Lattice-Based E-Cash, Revisited

Amit Deo[1,3], Benoît Libert[2,1], Khoa Nguyen[4], and Olivier Sanders[5]

[1] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, Inria, UCBL), France
[2] CNRS, Laboratoire LIP, France
[3] Inria, France
[4] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
[5] Orange Labs, Applied Crypto Group, Cesson-Sévigné, France

Abstract. Electronic cash (e-cash) was introduced 40 years ago as the digital analogue of traditional cash. It allows users to withdraw electronic coins that can be spent anonymously with merchants. As advocated by Camenisch *et al.* (Eurocrypt 2005), it should be possible to store the withdrawn coins compactly (i.e., with logarithmic cost in the total number of coins), which has led to the notion of *compact* e-cash. Many solutions were proposed for this problem but the security proofs of most of them were invalidated by a very recent paper by Bourse *et al.* (Asiacrypt 2019). The same paper describes a generic way of fixing existing constructions/proofs but concrete instantiations of this patch are currently unknown in some settings. In particular, compact e-cash is no longer known to exist under quantum-safe assumptions. In this work, we resolve this problem by proposing the first secure compact e-cash system based on lattices following the result from Bourse *et al.* Contrarily to the latter work, our construction is not only generic, but we describe two concrete instantiations. We depart from previous frameworks of e-cash systems by leveraging lossy trapdoor functions to construct our coins. The indistinguishability of lossy and injective keys allows us to avoid the very strong requirements on the involved pseudo-random functions that were necessary to instantiate the generic patch proposed by Bourse *et al.*

Keywords. Lattice-based cryptography, e-cash, anonymity, exculpability, provable security.

1 Introduction

The last decades have witnessed major changes in consumer habits, with a gradual shift to credit/debit cards for payments. Since 2016, the total amount of card payment transactions worldwide has indeed exceeded that of cash transactions,[6] as card transactions simply make spending easier and enable online purchases.

However, the benefits of electronic payments come at a price. Each transaction indeed leaks very sensitive information (at least to the entity managing the payment system), such as the identity of the recipient, the amount, the location

[6] https://avpsolutions.com/blog/payment-cards-now-set-to-surpass-cash/

of the spender, etc. For example, a patient paying his cardiologist with his card implicitly reveals to his bank that he probably has a heart condition, which is far from insignificant.

One could argue that, in some cases, the users' or recipients' identities can be masked through pseudonyms, but the concrete privacy benefits of this solution are questionable. Indeed, even for systems without central authority such as Bitcoin, pseudonymity only provides limited anonymity guarantees as shown for example by Ron and Shamir [32]. A natural question in this context is whether we can achieve the best of the two worlds. Namely, can we combine the features of electronic payments together with the anonymity of traditional cash?

Related Work. A first answer to this question was provided by Chaum in 1982 [13] when he introduced the notion of electronic cash (e-cash). Concretely, an electronic coin is the digital analogue of a standard coin/banknote that is issued by an authority, called a bank, to users. The authenticity of coins can be checked publicly, which allows users to spend them anonymously with any merchant who knows the bank public key. Unfortunately, the comparison stops there, as there is a major difference between physical and electronic coins. In the first case, the physical support is assumed to be unclonable, unless for extremely powerful adversaries. Obviously, the same assumption does not hold for digital data, and it is thus necessary to deter multiple spendings of the same coin.

However, detecting multiple uses of the same coin without affecting the anonymity of honest users is challenging. Chaum achieved this using blind signatures [13], by associating each coin with a serial number that remains hidden until the coin is spent. At this time, the serial number is added to a register that can be public, preluding crypto-currency ledgers. Using this register, anyone can detect the reuse of a coin, which leads to two families of e-cash systems.

The first one allows detecting frauds but does not enable the identification of perpetrators. In this case, detection must be performed before accepting payments. These systems are thus inherently *online*, as any recipient must be able to check the ledger at any time. This entails incompressible latencies to process payments that can be prohibitive in some situations, such as payments at tollgates, or at turnstiles for public transport.

The second family allows for the identification of defrauders. In this case, it is no longer necessary to check the coin upfront, as the defrauders know that they will ultimately be identified and then prosecuted. This simplifies the whole payment process as the e-cash system can now work *offline*. In 2020, the ability to process transactions offline might seem less appealing but it is still necessary today as mentioned in recent Visa [33] or Mastercard [28] technical documentations. Indeed, offline payments are inevitable in situations with no (or very limited) internet connections (aboard a plane, a boat, etc) and are still preferred in some regions. For example, a study by the french central bank [15] shows that, in 2013, less than 10 % of credit/debit cards in use in France are exclusively online (such cards being usually attributed to financially fragile persons); for the other cards, online checks are only performed on a random basis of for relatively

large amounts.

It is hard today to discuss e-cash without mentioning crypto-currencies such as Bitcoin, or even post-quantum ones such as MatRiCT [19]. The distinction between the two families above highlights the first difference between such systems. Crypto-currencies are indeed necessarily online whereas e-cash can be offline. However the main difference between these two systems rather lies in the trust model. The main strength of crypto-currencies is probably the absence of a central authority. This helps them circumvent the traditional reluctance of banks to novelties because a crypto-currency can be launched (almost) from scratch. In contrast, an e-cash system requires the support of a financial institution. Nevertheless, the absence of a central authority is also the main drawback of crypto-currencies. It indeed means that, in case of theft or loss of secret keys, the users lose everything, which is a major issue that we believe to be unacceptable for the general public. In the e-cash setting, where some authority manages the system, handling these situations is quite easy (corresponding procedures already exist for current payments systems such as debit/credit cards). There are also differences such as compliance with legislation, etc. In all cases, the very different features of both systems mean that they cannot be opposed. The reality is in fact the opposite and we should rather see crypto-currencies and e-cash systems as complementary solutions for privacy-preserving payments. From now on, we will only consider offline e-cash.

Following Chaum's seminal work, blind signatures were the first cornerstone of e-cash systems. Unfortunately, this design strategy comes with some limitations, such as the need to withdraw and store coins one by one, which quickly becomes cumbersome (see, e.g., [9]). This problem was addressed by Camenisch *et al.* [11] who proposed the notion of *compact* e-cash, where users withdraw and store N coins (that constitute a wallet) with constant, or at least logarithmic, complexity. The core idea of their construction – which has become the blueprint of most following works – is to associate each wallet with two seeds k and t for a pseudo-random function (PRF) family. These two seeds are then used to generate N pairs of pseudo-random values $(\mathsf{PRF}_k(i), \mathsf{PRF}_t(i))$. The former (i.e., $\mathsf{PRF}_k(i)$) serves as the coin serial number whereas $\mathsf{PRF}_t(j)$ essentially acts as a one-time pad on the spender's identity, resulting in a so-called double-spending tag. In case a coin is spent more than once, the same mask $\mathsf{PRF}_t(j)$ is used twice and can thus be cancelled out to reveal the defrauder's identity.

This elegant construction underlies many subsequent systems, including a candidate based on lattices [26]. Unfortunately, a recent result by Bourse *et al.* [9] has shown the limitations of this framework. In particular, they highlighted that systems based on it may fail to provably achieve exculpability, i.e., the property that honest users cannot be wrongly accused of double-spending a coin, even when the bank and other users conspire against them. As this issue underlies most of our design choices, we need to recall some details on it.

In the CHL construction [11], the serial number and the double-spending tag are constructed from the PRF outputs mentioned above but also from the spender's public key and some public data that can be seen as the unique iden-

tifier of a transaction. In case of double-spendings, it can be shown that the perpetrator will necessarily be identified. Unfortunately, Bourse *et al* pointed out that the opposite is not true, except in some very specific settings excluding lattices, as carefully crafted serial numbers and double-spending tags might lead the identification process to output a public key that was not even involved in the fraudulent transactions. Actually, two spendings from different users may even be considered as a double-spending by the system. As a consequence, the security proofs of the e-cash construction of Libert *et al.* [26] and of a subsequent improvement by Yang *et al.* [35] (the only schemes relying on quantum-resistant computational assumptions) are invalid and there is no known simple fix.

Before accusing a user, it is therefore necessary to perform additional verifications on the serial number occurring in a double-spending, in particular to ensure that it was constructed from the same seed and the same identity. This unfortunately seems infeasible given only $\mathsf{PRF}_k(i)$, as in [11]. To overcome this problem, the authors of [9] extended the serial number with new elements, each one being protected by a fresh PRF output. To ensure exculpability, it is then necessary to exclude collisions that could result from the PRF, leading to strong and non-standard requirements on the latter in [9]. Indeed, Bourse *et al.* need a notion of collision-resistance where, given the public parameters of a PRF family, the adversary should be unable to output two seeds k, k' and inputs x, x' such that $\mathsf{PRF}_k(x) = \mathsf{PRF}_{k'}(x')$. This might seem achievable by using a PRF based on symmetric primitives or by applying the techniques of Farshim *et al.* [20] to key-injective PRFs [24]. However, this would result in extremely inefficient e-cash constructions. Indeed, achieving security against cheating spenders requires to have them prove in zero-knowledge that they behaved honestly and correctly evaluated these PRFs, using certified seeds with valid inputs, etc. Such complex relations hardly coexist with the two solutions above. In particular, the Kim-Wu PRF [24] relies on a complex encoding of inputs into matrices which is hardly compatible with zero-knowledge techniques in lattices (recall that the PRF inputs should be part of witnesses). These rather require PRFs with a simpler algebraic structure, in the spirit of [4,7,3]. Unfortunately, the latter are not known to achieve collision-resistance. As of today, instantiating the Bourse *et al.* framework [9] from lattices would thus require to translate all statements to be proved into Boolean circuits. This would be much more expensive (by several orders of magnitude) than what we can hope for by leveraging the most efficient zero-knowledge techniques in standard lattices [35,8].

Our Contribution. In this paper, we show that we can dispense with the need for strong collision-resistance requirements by significantly departing from previous frameworks [11,9]. Our core idea is to perform only one standard PRF evaluation and use the resulting output to mask all the components of the serial number and double-spending tag, thanks to the use of a lossy trapdoor function F_{LTF} [30]. Recall that these are function families where injective evaluation keys are computationally indistinguishable from lossy evaluation keys, for which image elements reveal very little information on their preimages. In our construc-

tion, during a spending, we reveal $F_{\mathsf{LTF}}(\mathsf{PRF}_k(i))$ instead of $\mathsf{PRF}_k(i)$ and then extract randomness from the remaining entropy of $\mathsf{PRF}_k(i)$ in order to mask the spender's public key. This masked public key constitutes the second part of our serial number. When F_{LTF} is set up in its lossy mode in the proof of anonymity, we can show that the resulting serial number is indistinguishable from random and does not leak any sensitive information on the spender. Moreover, as F_{LTF} can be generated in injective mode in the real scheme, in case of colliding serial numbers, we are guaranteed that the same value $\mathsf{PRF}_k(i)$ is used in all the corresponding transactions. Together with the equality of serial numbers, this implies that the involved public keys necessarily coincide.

At this stage, we are ensured that a double-spending alert can only be generated by two transactions involving the same user. Then, it only remains to adapt the same technique to our double-spending tags, which is fairly simple. We can then prove security of our construction based only on the standard security properties of the pseudo-random function and the lossy trapdoor function.

However, as we intend to provide concrete constructions and not just frameworks, we still have to realise the missing component of the coin, namely the non-interactive zero-knowledge (NIZK) proofs that both the serial number and the double-spending tag are well-formed. Indeed, NIZK proofs are notoriously hard to produce in the lattice setting, at least compared to their counterparts in cyclic groups. We start from a very recent result by Yang *et al.* [35] which provides a protocol capturing many interesting lattice-related relations and show that it can be used to prove the statements required by our e-cash system. This is far from trivial as, in particular, spenders need to prove their correct composed evaluation of a pseudo-random function and a lossy trapdoor function using different parameters for the two primitives. We nevertheless manage to propose such NIZK arguments for two different PRF constructions [4,7], leading to two different instantiations. Notably, the GGM-based PRF [23] of Banerjee *et al.* [4] allows for the use of a polynomial modulus.

However, despite this nice asymptotic complexity, one should keep realistic expectations about the concrete performances of our scheme according to the current lattices state-of-the-art. We indeed note that, as of writing, most of our building blocks (zero-knowledge proofs, PRFs, etc) remain complex tools that can hardly compete with their pairing-based counterparts. This is highlighted by the recent paper by Yang et al [35] showing that existing (insecure) lattice e-cash constructions [26,35], which use building blocks similar to ours, generate transactions ranging from 260 MB to 720 TB. Fortunately, any future improvements of these tools could easily be leveraged by our construction. This is particularly true for our zero-knowledge proofs that we manage to represent as a standard instance of the powerful framework from [35].

Eventually, we propose the first concrete e-cash systems based on quantum-resistant hardness assumptions, following the reset of the state-of-the art resulting from [9]. Unlike [9] that modifies the CHL framework [11] by requiring stronger properties on the original building blocks, we upgrade it by considering alternative building blocks that are instantiable from standard lattice assump-

tions. Our work does not only lead to concrete constructions, but it also sheds new lights on e-cash by implicitly relying on a new framework which differs from [11,9] and does not require PRFs with non-standard security properties.

2 Preliminaries

We use lower-case bold characters (e.g. $\mathbf{x}$) to denote vectors and upper-case bold characters (e.g. $\mathbf{M}$) to denote matrices. The $(n \times n)$ identity matrix is denoted by $\mathbf{I}_n$. A superscript $\top$ for a vector or matrix denotes its transpose (e.g. $\mathbf{M}^\top$ is the transpose of $\mathbf{M}$). For any integer $q > 0$, $\mathbb{Z}_q$ denotes the integers modulo q. For integers $a < b$, $[a, b]$ denotes the set $\{a, a+1, \ldots, b\}$. Alternatively if $b > 1$, we define $[b] := \{1, \ldots, b\}$. For any real x, we denote by $\lfloor x \rfloor$ the greatest integer smaller than or equal to x. In addition, for positive integers n, p, q such that $q > p$, we define the rounding operation $\lfloor \cdot \rceil_p : \mathbb{Z}_q^n \to \mathbb{Z}_p^n$ as $\lfloor \mathbf{x} \rceil_p := \lfloor (p/q) \cdot \mathbf{x} \rfloor$. For probability distribution $\mathcal{D}$, we write $s \hookleftarrow \mathcal{D}$ to denote that s is a sample of the distribution $\mathcal{D}$. If X is a set, then $s \hookleftarrow U(X)$ represents the sampling of a uniform element of X. We also define the min-entropy of a discrete distribution $\mathcal{D}$ as $\mathsf{H}_\infty(\mathcal{D}) := -\log \max_{x'} \Pr_{x \hookleftarrow \mathcal{D}}[x = x']$. The statistical distance between two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ is denoted $\Delta(\mathcal{D}_1, \mathcal{D}_2)$. Throughout, we let λ denote a security parameter and use standard asymptotic notation $\mathcal{O}, \Theta, \Omega, \omega$ etc. We also use the standard notion of a pseudo-random function (PRF) and a zero-knowledge argument of knowledge (ZKAoK).

Binary decompositions. We use the same decompositions as those in [26] as explained next. Firstly, for any positive integer B and $\delta_B := \lfloor \log(B) \rfloor + 1$, we define the sequence $B_1, \ldots, B_{\delta_B}$ where $B_j := \lfloor \frac{B + 2^{j-1}}{2^j} \rfloor$ for $j \in [1, \delta_B]$. It can be verified that $\sum_{j \in [1, \delta_B]} B_j = B$. For any integer $x \in [0, B]$, there is an efficiently computable deterministic function $\mathsf{idec}_B : [0, B] \to \{0,1\}^{\delta_B}$ outputting a vector $\mathsf{idec}_B(x) =: \mathbf{y} \in \{0,1\}^{\delta_B}$ satisfying $\sum_{j \in [1, \delta_B]} B_j \cdot y_j = x$. The function idec_B can be extended to handle vector inputs, resulting in $\mathsf{vdec}_{m,B} : [0, B]^m \to \{0,1\}^{m \cdot \delta_B}$, for any integer $m > 0$. Explicitly, for any $\mathbf{x} \in [0, B]^m$, $\mathsf{vdec}_{m,B}(\mathbf{x}) := (\mathsf{idec}_B(x_1)^\top, \ldots, \mathsf{idec}_B(x_m)^\top)^\top$. In order to invert $\mathsf{vdec}_{m,B}$, we define the matrix $\mathbf{H}_{m,B} := (B_1, \ldots, B_{\delta_B}) \otimes \mathbf{I}_m$. It is easy to see that $\mathbf{H}_{m,B} \cdot \mathsf{vdec}_{m,B}(\mathbf{y}) = \mathbf{x}$. In addition, for any $x \in [0, B]$, we denote by $\mathsf{ibin}_B(x)$ the *standard* binary decomposition of x that fits into $\lfloor \log(B) \rfloor + 1$ bits. We define the binary representation of a vector to be the concatenation of the binary representations of its entries. Concretely, for any vector $\mathbf{x} \in [0, B]^m$, we define its binary representation to be $\mathsf{bin}_B(\mathbf{x})^\top := (\mathsf{ibin}(x_1), \ldots, \mathsf{ibin}(x_m))$.

2.1 Lattice Preliminaries

An m-dimensional lattice is a discrete subgroup of $\mathbb{R}^m$. For any integers n and q, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$ we define the full-rank lattice $\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = 0 \bmod q\}$ and the lattice coset $\Lambda_q^{\mathbf{u}}(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{u} \bmod q\}$. Defining $\rho_\sigma : \mathbb{R}^m \to \mathbb{R}$ as $\rho_\sigma(\mathbf{x}) := \exp(-\pi \|\mathbf{x}\|^2 / \sigma^2)$, the discrete Gaussian

distribution over a lattice coset L with parameter σ (denoted as $D_{L,\sigma}$) is the distribution with support L and mass function proportional to ρ_σ.

Hardness Assumptions. We will be assuming that both the learning with errors (LWE) and short integer solution (SIS) problems (as defined next) are hard for appropriate parameter settings.

Definition 1. *Let $m, n, q \in \mathbb{N}$ with $m > n$ and $\beta > 0$. The short integer solution problem $\mathsf{SIS}_{n,m,q,\beta}$ is, given $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n\times m})$, find a non-zero $\mathbf{x} \in \Lambda_q^\perp(\mathbf{A})$ with $0 < \|\mathbf{x}\| \leq \beta$.*

Definition 2. *Let q, α be functions of a parameter n. For a secret $\mathbf{s} \in \mathbb{Z}_q^n$, the distribution $A_{q,\alpha,\mathbf{s}}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is obtained by sampling $\mathbf{a} \hookleftarrow U(\mathbb{Z}_q^n)$ and a noise $e \hookleftarrow D_{\mathbb{Z},\alpha q}$, and returning $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e)$. The learning with errors problem $\mathsf{LWE}_{n,m,q,\alpha}$ is, for $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$, to distinguish between m independent samples from $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$ and the same number of samples from $A_{q,\alpha,\mathbf{s}}$.*

If m is omitted in the LWE problem, it is assumed that $m = \mathsf{poly}(n)$. If $q \geq \beta n^\delta$ for any constant $\delta > 0$ and $m, \beta = \mathsf{poly}(n)$, then standard worst-case lattice problems with approximation factors $\gamma = \max\{1, \beta^2/q\} \cdot \tilde{\mathcal{O}}(\beta\sqrt{n})$ reduce to $\mathsf{SIS}_{n,m,q,\beta}$ [29]. Alternatively, if $q \geq \sqrt{n}\beta$ and $m, \beta = \mathsf{poly}(n)$, then standard worst-case lattice problems with approximation factors $\gamma = \mathcal{O}(\beta\sqrt{n})$ reduce to $\mathsf{SIS}_{m,q,\beta}$ (see, e.g., [22, Sec. 9]). Similarly, if $\alpha q = \Omega(\sqrt{n})$, standard worst-case lattice problems with approximation factors $\gamma = \tilde{\mathcal{O}}(n/\alpha)$ reduce to $\mathsf{LWE}_{n,q,\alpha}$[31,10].

2.2 Lossy Trapdoor Functions

We will be using the notion of lossy trapdoor function (LTF) families from [30]. Informally, a lossy trapdoor function family can be used in one of two modes: a lossy mode and an injective mode. In the lossy mode, functions lose information on their inputs and cannot be inverted whereas in the injective mode, a trapdoor enables efficient inversion. In addition, there are generation algorithms that sample functions in either the lossy or injective mode. A crucial requirement is that no efficient adversary can distinguish whether a generation algorithm is outputting lossy functions or injective functions. We now recall the formal syntax and definition of an LTF family.

Definition 3. *An (m, k) lossy trapdoor function family with security parameter λ is a 4-tuple of PPT algorithms $(\mathsf{G}_0, \mathsf{G}_1, \mathsf{F}, \mathsf{F}^{-1})$ such that:*

- **(Injective Mode)** *$\mathsf{G}_0(1^\lambda)$ outputs a function index u and trapdoor τ. For any pair (u, τ) output by G_0, $\mathsf{F}(u, \cdot)$ computes an injective function $f_u : \{0,1\}^m \to \{0,1\}^*$ and $\mathsf{F}^{-1}(\tau, \mathsf{F}(u, x)) = x$.*
- **(Lossy Mode)** *$\mathsf{G}_1(1^\lambda)$ outputs a function index u. For any u output by G_1, $\mathsf{F}(u, \cdot)$ computes a lossy function $f_u : \{0,1\}^m \to \{0,1\}^*$, whose image is of size at most 2^{m-k}.*
- **(Indistinguishability)** *Let $(u, \tau) \leftarrow \mathsf{G}_0(1^\lambda)$ and $u' \leftarrow \mathsf{G}_1(1^\lambda)$. Then the distributions of u and u' are computationally indistinguishable.*

We will use the algorithms of the LTF family given in [30]. This family was reconstructed by Wee [34] where n, m, q, α are functions of λ, $p \leq q/(4n)$ and $\bar{n} = m/\log p$. In the following, $\mathbf{G} \in \mathbb{Z}_q^{m \times \bar{n}}$ is a special public matrix that allows to efficiently solve the bounded error decoding problem [30].

- $\mathsf{G}_0(n, m, q, \alpha)$: Sample $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m}), \mathbf{S} \hookleftarrow U(\mathbb{Z}_q^{n \times \bar{n}}), \mathbf{E} \hookleftarrow (\bar{\Psi}_\alpha)^{m \times \bar{n}}$ and output the index $\left(\mathbf{A}, \mathbf{B} := \mathbf{S}^\top \mathbf{A} + \mathbf{E}^\top + \mathbf{G}^\top\right)$ along with trapdoor $\mathbf{S}$.
- $\mathsf{G}_1(n, m, q, \beta)$: Sample $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m}), \mathbf{S} \hookleftarrow U(\mathbb{Z}_q^{n \times \bar{n}}), \mathbf{E} \hookleftarrow (\bar{\Psi}_\alpha)^{m \times \bar{n}}$ and output the index $\left(\mathbf{A}, \mathbf{B} := \mathbf{S}^\top \mathbf{A} + \mathbf{E}^\top\right)$
- F : On input $((\mathbf{A}, \mathbf{B}), \mathbf{x})$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{B} \in \mathbb{Z}_q^{\bar{n} \times m}$ and $\mathbf{x} \in \{0,1\}^m$, output $(\mathbf{A}\mathbf{x}, \mathbf{B}\mathbf{x})$
- F^{-1} : On input $(\mathbf{S}, (\mathbf{y}_1, \mathbf{y}_2))$ where $\mathbf{S} \in \mathbb{Z}_q^{n \times \bar{n}}, \mathbf{y}_1 \in \mathbb{Z}_q^n$ and $\mathbf{y}_2 \in \mathbb{Z}_q^{\bar{n}}$, compute $\mathbf{y} := \mathbf{y}_2 - \mathbf{S}^\top \mathbf{y}_1$. Use the efficient bounded-error decoder with respect to $\mathbf{G}$ on $\mathbf{y}$ to recover a vector $\mathbf{x}^* \in \{0,1\}^m$ such that $\mathbf{e}^* + \mathbf{G}^\top \mathbf{x}^* = \mathbf{y}$ for some small $\mathbf{e}^*$ with $\|\mathbf{e}^*\|_\infty \leq q/p$. Output $\mathbf{x}^*$.

Lemma 1 ([34]). *For any constant $\gamma < 1$ and n, take $q = \Theta(n^{1+1/\gamma})$, $p = \Theta(n^{1/\gamma})$ such that $p \leq q/(4n)$. Further, take $m = \mathcal{O}(n \log q), \alpha = \Theta(\sqrt{n}/q)$ and $\bar{n} = m/\log p$. Assuming that the $\mathsf{LWE}_{n,m,q,\alpha}$ problem is hard, the above construction is an (m, k)-LTF family where $k = (1-\gamma)m - n \log q$.*

The following instantiation of the generalized Leftover Hash Lemma of [17, Lemma 2.4] will be particularly useful:

Lemma 2. *Choose γ, n, q, p, α as in Lemma 1, arbitrary integers $n', q' > 2$ and an arbitrary distribution $\mathcal{X}$ over $\{0,1\}^m$. Then, for $\mathbf{A} \hookleftarrow U(\mathbb{Z}_{q'}^{n' \times m})$, $(\bar{\mathbf{A}}, \bar{\mathbf{B}}) \hookleftarrow \mathsf{G}_1(n, m, q, \alpha)$, $\mathbf{x} \hookleftarrow U(\mathcal{X})$ and $\mathbf{u} \hookleftarrow U(\mathbb{Z}_{q'}^{n'})$, we have*

$$\Delta\left(\left(\mathbf{A}\mathbf{x}, \mathbf{A}, (\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\mathbf{A}}\mathbf{x}, \bar{\mathbf{B}}\mathbf{x})\right), \left(\mathbf{u}, \mathbf{A}, (\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\mathbf{A}}\mathbf{x}, \bar{\mathbf{B}}\mathbf{x})\right)\right) \leq \frac{1}{2} \cdot \sqrt{2^{-(\mathsf{H}_\infty(\mathcal{X}) - (m\gamma + n \log q + n' \log q'))}}.$$

2.3 Witness Extraction and Forking Lemma

Recall that the transcript of a Σ-protocol consists of three messages starting with a message from a prover to a verifier. The Fiat-Shamir transform [21] provides a well-known method to remove interaction from a Σ-protocol. In particular, the second message (which is a uniformly chosen "challenge" value from the verifier to the prover) is replaced by the evaluation of a random oracle on input given by the first message. When adopting this method, it is important to carefully argue that the resulting non-interactive protocol is still an argument of knowledge. That is, if a prover convinces the verifier to accept with non-negligible probability, then replaying the prover allows for the extraction of a witness to the statement in question. This is usually achieved by applying a "forking lemma".

We will focus on the argument system of Yang et al.[35] which takes the three-message form of a Σ-protocol. The witness extraction for the interactive

ZKAoK of Yang et al. requires any $\ell = 3$ accepting transcripts, all with the same first prover message but distinct challenge values. We refer to ℓ such accepting transcripts as an ℓ-fork.

When using a random oracle to remove interaction with our chosen argument system, a forking lemma that considers the probability of producing an ℓ-fork for $\ell = 3$ should be used. The extended/generalised forking lemma of El Kaafarani and Katsumata [18, Lemma 1] provides a forking lemma for any $\ell \geq 2$. For simplicity, we state their result in the special case that $\ell = 3$.

Lemma 3 ([18]). *Fix some input $x \in \{0,1\}^*$ and take some arbitrary set* accept*. Let $\mathcal{A}$ be an efficient algorithm outputting triples (m_1, m_2, m_3) on input x that has oracle access to a random oracle $H : \{0,1\}^* \to [h]$ and let Q be an upper bound on the number of queries that $\mathcal{A}$ makes to H. Denote*

$$\mathsf{acc} := \Pr\left[(m_1, m_2, m_3) \leftarrow \mathcal{A}^{H(\cdot)}(x) : \begin{smallmatrix}(m_1,m_2,m_3) \in \mathsf{accept} \wedge \\ m_2 \text{ is the result of an } H\text{-query}\end{smallmatrix}\right]$$

$$\mathsf{frk}_3 := \Pr\left[((m_1, m_{2,i}, m_{3,i}))_{i=1}^3 \leftarrow F^{\mathcal{A}}(x) : \begin{smallmatrix}\forall i \in \{1,2,3\} : (m_1,m_{2,i},m_{3,i}) \in \mathsf{accept} \\ \wedge\ (m_{2,i})_{i=1}^3 \text{are pairwise distinct}\end{smallmatrix}\right]$$

for any efficient algorithm $F^{\mathcal{A}}$ that runs $\mathcal{A}$ at most 3 times. Then, for a particular choice of $F^{\mathcal{A}}$,

$$\mathsf{frk}_3 \geq \mathsf{acc} \cdot \left(\left(\frac{\mathsf{acc}}{Q}\right)^2 - \frac{3}{h}\right).$$

2.4 E-Cash Security Definitions

E-cash systems involve three types of parties: banks denoted $\mathcal{B}$, users denoted $\mathcal{U}$ and merchants denoted $\mathcal{M}$. The syntax of an offline compact e-cash system consists of the following algorithms/protocols:

ParGen$(1^\lambda, 1^L)$**:** On input a security parameter λ and wallet size $L = \log(\mathsf{poly}(\lambda))$, outputs public parameters par containing L (amongst other things).

BKeyGen$(1^\lambda, \mathsf{par})$**:** On input par, outputs a key pair $(PK_{\mathcal{B}}, SK_{\mathcal{B}})$ for the bank, which allows $\mathcal{B}$ to issue wallets of size 2^L.

UKeyGen $(1^\lambda, \mathsf{par})$**:** On input par, generates a key pair $(PK_{\mathcal{U}}, SK_{\mathcal{U}})$ for the user.

MKeyGen$(1^\lambda, \mathsf{par})$**:** On input par, generates $(PK_{\mathcal{M}}, SK_{\mathcal{M}})$ for the merchant.

We henceforth assume that all algorithms implicitly take par as input.

Withdraw $(\mathcal{U}(PK_{\mathcal{B}}, SK_{\mathcal{U}}), \mathcal{B}(PK_{\mathcal{U}}, SK_{\mathcal{B}}))$**:** An interactive protocol that allows $\mathcal{U}$ to obtain a wallet $\mathcal{W}$ consisting of 2^L coins or an error message $\perp$. The bank $\mathcal{B}$ obtains tracing information $\mathsf{T}_{\mathcal{W}}$.

Spend $(\mathcal{U}(\mathcal{W}, PK_{\mathcal{B}}, PK_{\mathcal{M}}), \mathcal{M}(SK_{\mathcal{M}}, PK_{\mathcal{U}}, PK_{\mathcal{B}}, \mathsf{info}))$**:** A protocol allowing a user $\mathcal{U}$ to give a coin from $\mathcal{W}$ to merchant $\mathcal{M}$ with respect to transaction metadata info. The user outputs an updated wallet $\mathcal{W}'$ whereas the output of $\mathcal{M}$ is a coin coin consisting of info, a serial number, a security tag and a proof of validity or an error symbol $\perp$.

VerifyCoin $(PK_{\mathcal{B}}, \mathsf{coin})$**:** Outputs 1 if the proof of validity in coin verifies correctly with respect to $PK_{\mathcal{B}}$ and 0 otherwise.

VerifyDeposit $(PK_{\mathcal{B}}, PK_{\mathcal{M}}, \mathsf{coin}, \mu)$**:** Outputs 1 if the proof of validity in coin verifies correctly with respect to $PK_{\mathcal{B}}$ and if the data μ verifies correctly with respect to $PK_{\mathcal{M}}$. Else, outputs 0.

Deposit $(\mathcal{M}(SK_{\mathcal{M}}, \mathsf{coin}, PK_{\mathcal{B}}), \mathcal{B}(PK_{\mathcal{M}}, SK_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}}))$**:** This is a protocol allowing $\mathcal{M}$ to deposit coin (containing some metadata info) in its account with $\mathcal{B}$. In the protocol, $\mathcal{M}$ sends coin along with some data μ. Then, $\mathcal{B}$ uses a list $\mathsf{state}_{\mathcal{B}}$ of previously deposited coins to proceed as follows. If $\mathsf{VerifyCoin}\,(PK_{\mathcal{B}}, \mathsf{coin}) = 0$ or $\mathsf{VerifyDeposit}\,(PK_{\mathcal{B}}, PK_{\mathcal{M}}, \mathsf{coin}, \mu) = 0$, $\mathcal{B}$ outputs $\perp$. If info and $PK_{\mathcal{M}}$ exist in the same entry of $\mathsf{state}_{\mathcal{B}}$, then $\mathcal{B}$ returns this entry $(\mathsf{coin}, PK_{\mathcal{M}}, \mu')$. If the serial number $\mathbf{y}_S$ derived from coin is not in $\mathsf{state}_{\mathcal{B}}$, then $\mathcal{B}$ adds the tuple $(\mathsf{coin}, PK_{\mathcal{M}}, \mu, \mathbf{y}_S)$ to $\mathsf{state}_{\mathcal{B}}$. If there is some tuple $(\mathsf{coin}', PK'_{\mathcal{M}}, \mu', \mathbf{y}_S)$ in $\mathsf{state}_{\mathcal{B}}$, then $\mathcal{B}$ outputs such a tuple.

Identify $(PK_{\mathcal{B}}, \mathsf{coin}_1, \mathsf{coin}_2)$**:** An algorithm allowing to identify a double spender $\mathcal{U}$ whenever coin_1 and coin_2 share the same serial number. The output of this algorithm is a public key $PK_{\mathcal{U}}$ and a proof that this public key corresponds to a double spender Π_G.

E-cash systems should provide the following properties whose formal definitions, adapted from [26,9], are provided below.

- **Anonymity:** no coalition of banks and merchants can identify the wallet that a coin originates from.
- **Traceability:** the bank is always able to identify at least one member of a coalition that has spent more than it has withdrawn. This property introduced by Canard *et al.* [12] simultaneously captures the balance and identification properties considered in [11,6].
- **Strong exculpability:** no coalition of banks and merchants can convincingly accuse an innocent user of double-spending.
- **Clearing:** an honest merchant is always able to deposit the received coins. In particular, no coalition of bank, merchants and users can generate a convincing proof that the latter have already been deposited.

Definition 4. *An e-cash system provides* **anonymity** *if there exists an efficient simulator* $\mathcal{S} = (\mathsf{SimParGen}, \mathsf{SimSpend})$ *such that no PPT adversary* $\mathcal{A}$ *has non-negligible advantage in the* ***anonymity*** *game described below:*

1. *The challenger flips a fair coin* $d \hookleftarrow U(\{0,1\})$ *and runs* $\mathsf{par} \leftarrow \mathsf{ParGen}(1^\lambda, 1^L)$ *if* $d = 1$ *and* $(\mathsf{par}, \tau_{sim}) \hookleftarrow \mathsf{SimParGen}(1^\lambda, 1^L)$ *otherwise. In either case, it gives* par *to* $\mathcal{A}$.
2. $\mathcal{A}$ *outputs some public key* $PK_{\mathcal{B}}$ *and adaptively invokes the following oracles:*
   - $\mathcal{Q}_{\mathsf{GetKey}}(i)$*: this oracle generates* $(SK_{\mathcal{U}_i}, PK_{\mathcal{U}_i}) \hookleftarrow \mathsf{UKeygen}(\mathsf{par})$ *if it does not exist yet and returns* $PK_{\mathcal{U}_i}$.
   - $\mathcal{Q}_{\mathsf{Withdraw}}(PK_{\mathcal{B}}, i)$*: this oracle plays the role of user* $\mathcal{U}_i$ *– and creates their key pair if it does not exist yet – in an execution of the withdrawal protocol* $\mathsf{Withdraw}\big(\mathcal{U}(\mathsf{par}, PK_{\mathcal{B}}, SK_{\mathcal{U}_i}), \mathcal{A}(\mathsf{state})\big)$*, with the adversary* $\mathcal{A}$ *playing the role of the bank. At the* j*-th query, we denote by* $\mathcal{W}_j$ *the user's output which may be a valid wallet or an error message* $\perp$.

- $\mathcal{Q}_{\mathsf{Spend}}(PK_{\mathcal{B}}, i, j, PK_{\mathcal{M}}, \mathsf{info})$*: the oracle first checks if the wallet* $\mathcal{W}_j$ *has been issued to* $\mathcal{U}_i$ *by the bank* $\mathcal{B}$ *via an invocation of* $\mathcal{Q}_{\mathsf{Withdraw}}(PK_{\mathcal{B}}, i)$*. If not, the oracle outputs* $\perp$*. Otherwise,* $\mathcal{Q}_{\mathsf{Spend}}$ *checks if the internal counter* J *of* $\mathcal{W}_j$ *satisfies* $J < 2^L - 1$*. If* $J = 2^L - 1$*, it outputs* $\perp$*. Otherwise,* $\mathcal{Q}_{\mathsf{Spend}}$ *responds as follows:*
  - *If* $d = 1$*, it runs* $\mathsf{Spend}\big(\mathcal{U}_i(\mathcal{W}_j, PK_{\mathcal{B}}, PK_{\mathcal{M}}), \mathcal{A}(\mathsf{state}, \mathsf{info})\big)$ *with the merchant-executing* $\mathcal{A}$ *in order to spend a coin from* $\mathcal{W}_j$*.*
  - *If* $d = 0$*,* $\mathcal{Q}_{\mathsf{Spend}}$ *runs* $\mathsf{SimSpend}\big(\mathsf{par}, \tau_{sim}, PK_{\mathcal{B}}, PK_{\mathcal{M}}, \mathsf{info}\big)$*.*

3. *When* $\mathcal{A}$ *halts, it outputs a bit* $d' \in \{0, 1\}$ *and wins if* $d' = d$*. The adversary's advantage is the distance* $\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{A}}(\lambda) := |\Pr[d' = d] - 1/2|$*, where the probability is taken over all coin tosses.*

Definition 5. *An e-cash system ensures* **traceability** *if, for any PPT adversary* $\mathcal{A}$*, the experiment below outputs* 1 *with negligible probability:*

1. *The challenger generates public parameters* $\mathsf{par} \hookleftarrow \mathsf{ParGen}(1^\lambda, 1^L)$ *and a public key* $(PK_{\mathcal{B}}, SK_{\mathcal{B}}) \hookleftarrow \mathsf{BKeyGen}(\mathsf{par})$*. It gives* par *and* $PK_{\mathcal{B}}$ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *is granted access to the oracle* $\mathcal{Q}_{\mathsf{Withdraw}}(PK_{\mathcal{U}})$ *that plays the role of the bank* $\mathcal{B}$ *in an execution of* $\mathsf{Withdraw}(\mathcal{A}(\mathsf{state}), \mathcal{B}(\mathsf{par}, PK_{\mathcal{U}}, SK_{\mathcal{B}}))$ *with* $\mathcal{A}$*, acting as a cheating user. After each query, the challenger stores in a database* T *the information* $\mathsf{T}_{\mathcal{W}} = PK_{\mathcal{U}}$*, or* $\perp$ *if the protocol fails.*
3. *After* Q_w *polynomially many queries,* $\mathcal{A}$ *outputs coins* $\{\mathsf{coin}_i\}_{i=1}^N$ *which are parsed as* $(\mathsf{info}_i, PK_{\mathcal{M}_i}, S_i, \pi_i)$*. The experiment returns 1, unless (at least) one of the following conditions holds (in which case, it returns 0):*
   - $N \leq 2^L \cdot Q_w$*;*
   - $\exists (i, j) \in \{1, \ldots, N\}^2$ *such that* $(\mathsf{info}_i, PK_{\mathcal{M}_i}) = (\mathsf{info}_j, PK_{\mathcal{M}_j})$*;*
   - $\exists i \in \{1, \ldots, N\}$ *such that* $\mathsf{VerifyCoin}(PK_{\mathcal{B}}, \mathsf{coin}_i) = 0$*;*
   - $\exists (i, j) \in \{1, \ldots, N\}^2$ *such that* $\mathsf{Identify}\big(\mathsf{par}, PK_{\mathcal{B}}, coin_i, coin_j\big)$ *returns a public key* $PK_{\mathcal{U}}$ *that belongs to the database* T*.*

Definition 6. *An e-cash system provides* **strong exculpability** *if no PPT adversary* $\mathcal{A}$ *has noticeable success probability in the game below:*

1. *The challenger runs* $\mathsf{par} \leftarrow \mathsf{ParGen}(1^\lambda, 1^L)$*, gives* par *to* $\mathcal{A}$ *and initializes empty sets of honest users* $\mathcal{HU}$*, wallets* T_{FW} *and double spent coins* T_{ds}*.*
2. $\mathcal{A}$ *generates* $PK_{\mathcal{B}}$ *on behalf of the bank and interacts with these oracles:*
   - $\mathcal{Q}_{\mathsf{GetKey}}(i)$*: this oracle generates* $(SK_{\mathcal{U}_i}, PK_{\mathcal{U}_i}) \hookleftarrow \mathsf{UKeygen}(\mathsf{par})$ *if it does not exist yet and returns* $PK_{\mathcal{U}_i}$*, which is added to* $\mathcal{HU}$*.*
   - $\mathcal{Q}_{\mathsf{Withdraw}}(PK_{\mathcal{B}}, i)$*: this oracle plays the role of* $\mathcal{U}_i$ *– and creates* $(SK_{\mathcal{U}_i}, PK_{\mathcal{U}_i})$ *if it does not exist yet – in an execution of* $\mathsf{Withdraw}\big(\mathcal{U}(\mathsf{par}, PK_{\mathcal{B}}, SK_{\mathcal{U}_i}), \mathcal{A}(\mathsf{state})\big)$ *where* $\mathcal{A}$ *plays the role of the bank. At the* j*-th such query, we denote by* $\mathcal{W}_j$ *the user's output. If the protocol succeeds* ($\mathcal{W}_j = \perp$)*, then* $(j, \mathcal{W}_j)$ *is added to* T_{FW}*.*
   - $\mathcal{Q}_{\mathsf{Spend}}(PK_{\mathcal{B}}, i, j, PK_{\mathcal{M}}, \mathsf{info})$*: the oracle first checks if the wallet* $\mathcal{W}_j$ *was provided to* $\mathcal{U}_i$ *via an invocation of* $\mathcal{Q}_{\mathsf{Withdraw}}(\mathsf{par}, PK_{\mathcal{B}}, i)$ *using* T_{FW}*. If not, the oracle outputs* $\perp$*. If the internal counter of* $\mathcal{W}_j$ *satisfies* $J = 2^\ell -$

1, *then* $\mathcal{W}_j$ *is reset to its original state, where* $J = 0$. *Then,* $\mathcal{Q}_{\mathit{Spend}}$ *spends a coin from* $\mathcal{W}_j$ *by running* $\mathsf{Spend}\big(\mathcal{U}_i(\mathcal{W}_j, PK_{\mathcal{B}}, PK_{\mathcal{M}}), \mathcal{A}(\mathsf{state}, \mathsf{info})\big)$ *with* $\mathcal{A}$. *If the resulting coin has the same serial number* S *as a previous query* $\mathcal{Q}_{\mathit{Spend}}\big(PK_{\mathcal{B}}, i, j, \cdot, \cdot\big)$ *then add* (i, j, S) *to* T_{ds}.

3. *When adversary* $\mathcal{A}$ *halts, it outputs two coins* $\mathsf{coin}_1, \mathsf{coin}_2$. *It is declared successful if* $\mathsf{Identify}(\mathsf{par}, PK_{\mathcal{B}}, \mathsf{coin}_1, \mathsf{coin}_2) \in \mathcal{HU}$ *and* $\forall (i, j), (i, j, S) \notin \mathsf{T}_{\mathsf{ds}}$ *where* S *is the common serial number shared by* coin_1 *and* coin_2

Definition 7. *An e-cash system ensures* **clearing** *if for any PPT adversary* $\mathcal{A}$, *the probability of* $\mathcal{A}$ *winning the* ***clearing*** *game below is negligible:*

1. *The challenger runs* $\mathsf{par} \leftarrow \mathsf{ParGen}(1^\lambda, 1^L)$, *gives* par *to* $\mathcal{A}$ *and initializes a set of honest merchants* $\mathcal{HM}$ *which is initially empty.*
2. $\mathcal{A}$ *generates* $PK_{\mathcal{B}}$ *on behalf of the bank and interacts with these oracles:*
   - $\mathcal{Q}_{\mathit{GetKey}}(i)$: *this oracle generates* $(SK_{\mathcal{M}_i}, PK_{\mathcal{M}_i}) \hookleftarrow \mathsf{MKeygen}(\mathsf{par})$ *if it does not exist yet and returns* $PK_{\mathcal{M}_i}$, *which is added in* $\mathcal{HM}$.
   - $\mathcal{Q}_{\mathit{Receive}}\big(PK_{\mathcal{B}}, i\big)$: *this oracle plays the role of a merchant – and creates* $(SK_{\mathcal{M}_i}, PK_{\mathcal{M}_i})$ *if it does not exist yet – in an execution of* $\mathsf{Spend}\big(\mathcal{A}(\mathsf{state}), \mathcal{M}_i(SK_{\mathcal{M}}, PK_{\mathcal{U}}, \mathsf{info})\big)$ *where* $\mathcal{A}$ *plays the role of the user. At the* j*-th query, we denote by* coin_j *the merchant's output.*
   - $\mathcal{Q}_{\mathit{Deposit}}\big(PK_{\mathcal{B}}, i, j\big)$: *this oracle plays the role of the merchant in an execution of* $\mathsf{Deposit}(\mathcal{M}_i(SK_{\mathcal{M}_i}, \mathsf{coin}_j, PK_{\mathcal{B}}), \mathcal{A}(\mathsf{state}))$ *with* $\mathcal{A}$ *playing the role of* $\mathcal{B}$. *It however aborts if* $PK_{\mathcal{M}_i} \notin \mathcal{HM}$, *if* coin_j *has not been received by merchant* i *or if it has already been deposited.*
3. *When* $\mathcal{A}$ *halts, it outputs a tuple* $(PK_{\mathcal{M}}, \mathsf{coin}, \mu)$. *The adversary wins if* $PK_{\mathcal{M}} \in \mathcal{HM}$, $\mathit{VerifyDeposit}\,(PK_{\mathcal{B}}, PK_{\mathcal{M}}, \mathsf{coin}, \mu) = 1$ *and* coin *has not been involved in a previous* $\mathcal{Q}_{\mathit{Deposit}}$ *query.*

3 Intuition

The core of an e-cash system is the pair constituted by the serial number $\mathbf{y}_S$ and the double-spending tag $\mathbf{y}_T$ of a coin. Besides zero-knowledge proofs, they are essentially the only elements made public during a spending and therefore must comply with very strong anonymity requirements while allowing the identification of double-spenders. In addition, it should be possible to (efficiently) prove that they are well-formed, which rules out most simple constructions.

Designing such elements is thus far from trivial which partially explains why most e-cash systems have followed the elegant idea proposed by Camenisch *et al* [11]. It relies on a pseudo-random function PRF as follows. For a wallet of $N = 2^L$ coins, a first seed $\mathbf{k}$ is used to generate N pseudo-random values $\mathsf{PRF}_{\mathbf{k}}(i)$, for $i \in [1, N]$, acting as the coins' serial numbers. Meanwhile, a second seed $\mathbf{t}$ allows generating independent values $\mathsf{PRF}_{\mathbf{t}}(i)$ acting as one-time pads on the spender's identity. The concealed identity constitutes the double-spending tag.

Any user can generate at most N fresh pairs $(\mathsf{PRF}_{\mathbf{k}}(i), \mathsf{PRF}_{\mathbf{t}}(i))$ per wallet. In case of double-spending, a pair must have been re-used, meaning that a serial

number $\mathsf{PRF}_{\mathbf{k}}(i)$ will appear twice in the bank database, thus making frauds easy to detect. Moreover, in such a case, the spender's identity will be masked using the same value $\mathsf{PRF}_{\mathbf{t}}(i)$. An appropriate combination of the corresponding double-spending tags thus allows to remove $\mathsf{PRF}_{\mathbf{t}}(i)$ and so to identify the defrauder. Some adjustments are necessary in the lattice setting [26], but the high-level principle remains the same.

However, Bourse *et al* [9] recently showed that this approach may fail to provide a sound proof of exculpability in many cases. Indeed, the identity returned by the identification algorithm is a complex mix of PRF outputs, public keys and random values, most of them being totally controlled by the adversary. It is therefore impossible to guarantee that the returned identity corresponds to the author of these fraudulent payments nor even to guarantee that both payments have been performed by the same user.

In [9], Bourse *et al.* point out that this problem is partially due to a misidentification of the properties that must be satisfied by the pseudo-random function PRF. They therefore propose to strengthen the requirements on PRF, introducing in particular a notion of collision resistance that essentially states the difficulty of finding $(\mathbf{s}, \mathbf{s}', i, i')$ such that $\mathsf{PRF}_{\mathbf{s}}(i) = \mathsf{PRF}_{\mathbf{s}'}(i')$. Assuming that the PRF satisfies such suitable properties, they prove security of generic constructions that are reminiscent of the seminal scheme proposed by Camenisch *et al.* An interesting aspect of [9] is thus the rehabilitation of the original intuition of compact e-cash [11] that has been common to all following works.

Unfortunately, this is done by relying on unconventional security notions for PRFs that have not been considered by designers of such functions. Bourse *et al.* show that, under suitable assumptions, these notions are actually already satisfied by some PRFs in cyclic groups, but similar results are not known in the lattice setting. Indeed, existing lattice-based PRFs are not known to provide collision-resistance in this strong sense, which prevents instantiation of their frameworks in this setting. Concretely, this means that secure lattice-based e-cash systems are not known to exist for the time being.

In this work, we choose a very different strategy that we believe to be better suited for the lattice setting as it does not rely on collision-resistant PRFs.

Our first step is to modify the construction of the serial numbers to ensure that collisions only occur for spendings performed by the same user. In [9], this is achieved by using the same seed (but different public parameters) to generate all the pseudo-random values used during a spending. Assuming that pseudo-randomness still holds in this context and that collision resistance is achieved by some of the PRFs, they prove that a collision only occurs for spendings involving the same seed. They are then able to prove that the use of the same seed implies the involvement of the same user, and so on until proving exculpability of their construction. Here, we still use a PRF as a source of pseudo-random values but our serial numbers are not simply the outputs of such a function. We indeed want to reuse the same pseudo-random values for different parts of our serial numbers and double-spending tags to rule out the adversarial strategy pointed out in [9]. To achieve this while retaining anonymity, we use the notion of a lossy trapdoor

function F_{LTF} introduced in [30] and more specifically, the elegant instantiation based on LWE proposed in [34] related to the original construction in [30].

The first element of $\mathbf{y}_S$ is now $F_{\mathsf{LTF}}(\mathsf{PRF}_{\mathbf{k}}(i))$, which still allows to extract random bits from $\mathsf{PRF}_{\mathbf{k}}(i)$ using a universal hash function H_{UH}, as proved in [30]. We can thus incorporate $PK_{\mathcal{U}} + H_{\mathsf{UH}}(\mathsf{PRF}_{\mathbf{k}}(i))$ in $\mathbf{y}_S$ while ensuring anonymity of the user $\mathcal{U}$ that owns the public key $PK_{\mathcal{U}}$. In the exculpability proof, we will generate F_{LTF} in the injective mode, thus ensuring that a collision $\mathbf{y}_S = \mathbf{y}_{S'}$ can only occur when the same value $\mathsf{PRF}_{\mathbf{k}}(i)$ is used for both transactions. Together with $PK_{\mathcal{U}} + H_{\mathsf{UH}}(\mathsf{PRF}_{\mathbf{k}}(i)) = PK_{\mathcal{U}'} + H_{\mathsf{UH}}(\mathsf{PRF}_{\mathbf{k}}(i))$, this implies $PK_{\mathcal{U}} = PK_{\mathcal{U}'}$.

We then adapt this idea to double-spending tags. We similarly extract random bits from $\mathsf{PRF}_{\mathbf{k}}(i)$ using a different universal hash function H'_{UH} to define $\mathbf{y}_T = PK_{\mathcal{U}} + \mathsf{FRD}(R) \cdot H'_{\mathsf{UH}}(\mathsf{PRF}_{\mathbf{k}}(i))$, where $\mathsf{FRD}(R)$ is some public matrix specific to the transaction. As $\mathsf{PRF}_{\mathbf{k}}(i)$ and the public key $PK_{\mathcal{U}}$ are the same for both transactions, the formula $\mathbf{y}_T - \mathsf{FRD}(R) \cdot [(\mathsf{FRD}(R) - \mathsf{FRD}(R'))^{-1} \cdot (\mathbf{y}_T - \mathbf{y}_{T'})]$ necessarily returns such a public key whose owner is guilty of double-spendings.

As far as efficiency goes, we essentially add some matrix-vector products to the construction of [26]. Moreover, since all of these matrices are public, a NIZK proof of correct computations can be produced using the framework provided in [26] or the more efficient techniques in Section 5.

4 Construction

We present a new e-cash system that overcomes the aforementioned issues in the proof of exculpability. We use the PRF from [7] that allows for a simpler description of our system. We nevertheless explain in Section 7 how to improve efficiency by using the alternative PRF from [4]. While the Withdraw protocol is a simplification of [26], the Spend protocol is very different in the way to construct coin serial numbers and security tags. Additional details on the zero-knowledge arguments of knowledge used in our construction are given in Section 5.

ParGen$(1^\lambda, 1^L)$**:** Given security parameter λ and integer $L > 0$ such that 2^L is the desired number of coins per wallet issued, perform the following:

1. Choose secure public parameters $\mathsf{par}_{\mathsf{PRF}} = \big(m, n, p, q, \mathbf{P}_0, \mathbf{P}_1\big)$ for the BLMR PRF family [7]. Namely,
   a. For $n = \mathcal{O}(\lambda)$, set $\alpha = 2^{-\omega(\log^{1+c}(n))}$ for some constant $c > 0$; a prime $p = 2^{\log^{1+c}(n)}$; a prime power $q = \mathcal{O}(\sqrt{n}/\alpha)$ such that p divides q; and $m = n \cdot \lceil \log q \rceil$.
   b. Sample $\mathbf{P}_0, \mathbf{P}_1 \hookleftarrow U(\{0,1\}^{m \times m})$ over $\mathbb{Z}_q$-invertible matrices.
2. Choose parameters $\mathsf{par}_{sig} = (q_s, \ell, \sigma, (m_i)_{i=0}^3, m_s, m_f)$ for a signature scheme allowing to sign committed values [25]. Namely,
   a. Choose a prime power modulus $q_s = \tilde{\mathcal{O}}(n^3)$ dividing q, an integer $\ell = \Theta(\lambda)$ and a Gaussian parameter $\sigma = \Omega(\sqrt{n \log q_s} \log n)$. Set $\delta_{q_s - 1} = \lceil \log_2(q_s) \rceil$, $\delta_{q-1} = \lceil \log_2(q) \rceil$ and $\delta_{p-1} = \lceil \log_2(p) \rceil$. Define the message block lengths $m_0 = m_s := 2n\delta_{q_s-1}$, as well as $m_1 = m$ and $m_2 = \bar{m} := m\delta_{q-1}$.

b. Sample $\mathbf{D}'_0, \mathbf{D}''_0 \hookleftarrow U(\mathbb{Z}_{q_s}^{n \times m_0})$ and $\mathbf{D}_i \hookleftarrow U(\mathbb{Z}_{q_s}^{n \times m_i})$, for $i \in \{1,2\}$, and define the commitment key to be $CK := (\mathbf{D}_0 := [\mathbf{D}'_0 | \mathbf{D}''_0], \mathbf{D}_1, \mathbf{D}_2)$.
c. Sample $\mathbf{F} \hookleftarrow U(\mathbb{Z}_p^{n \times m})$.

3. Choose parameters $\mathsf{par}_{\mathsf{LTF}}$ for the lossy trapdoor function of [30]. In terms of $n_{\mathsf{LTF}} = \tilde{\mathcal{O}}(\lambda)$ for constant $c > 0$, these consist of moduli $q_{\mathsf{LTF}} = \Theta(n_{\mathsf{LTF}}^{1+1/\gamma})$ that divides q and $p_{\mathsf{LTF}} = \Theta(n_{\mathsf{LTF}}^{1/\gamma})$ for some constant $\gamma < 1$; matrix dimensions n_{LTF} and $m_{\mathsf{LTF}} = \Theta(n_{\mathsf{LTF}} \log q_{\mathsf{LTF}})$ and $\bar{n}_{\mathsf{LTF}} = \bar{m}_{\mathsf{LTF}} / \log p_{\mathsf{LTF}}$ such that $p_{\mathsf{LTF}} < q_{\mathsf{LTF}}/4n_{\mathsf{LTF}}$; and an LWE error rate $\alpha_{\mathsf{LTF}} = \Theta(\sqrt{n}/q_{\mathsf{LTF}})$. We additionally require that $m_{\mathsf{LTF}} = m \cdot \lceil \log p \rceil$. Then, select an evaluation key ek_{LTF} for a lossy trapdoor function in injective mode $F_{\mathsf{LTF}} : \{0,1\}^{m_{\mathsf{LTF}}} \to \mathbb{Z}_{q_{\mathsf{LTF}}}^{n_{\mathsf{LTF}} + \bar{n}_{\mathsf{LTF}}}$, meaning that $ek_{\mathsf{LTF}} = \left(\mathbf{A}_{\mathsf{LTF}}, \mathbf{U}_{\mathsf{LTF}}\right)$ consists of a random $\mathbf{A}_{\mathsf{LTF}} \hookleftarrow U(\mathbb{Z}_q^{n_{\mathsf{LTF}} \times m_{\mathsf{LTF}}})$ and a matrix

$$\mathbf{U}_{\mathsf{LTF}} = \mathbf{S}_{\mathsf{LTF}}^{\top} \cdot \mathbf{A}_{\mathsf{LTF}} + \mathbf{E}_{\mathsf{LTF}}^{\top} + \mathbf{G}_{\mathsf{LTF}}^{\top} \in \mathbb{Z}_{q_{\mathsf{LTF}}}^{\bar{n}_{\mathsf{LTF}} \times m_{\mathsf{LTF}}},$$

for some $\mathbf{S}_{\mathsf{LTF}} \hookleftarrow U(\mathbb{Z}_{q_{\mathsf{LTF}}}^{n_{\mathsf{LTF}} \times \bar{n}_{\mathsf{LTF}}})$, $\mathbf{E}_{\mathsf{LTF}} \hookleftarrow D_{\mathbb{Z}^{m_{\mathsf{LTF}} \times \bar{n}_{\mathsf{LTF}}}, \alpha_{\mathsf{LTF}} q_{\mathsf{LTF}}}$ and $\mathbf{G}_{\mathsf{LTF}}$ referred to in the preliminaries.

4. Choose an integer $\bar{p} > 0$ such that $\bar{p} < p/2$ which will define a challenge space $\{-\bar{p}, \ldots, \bar{p}\}$ for the argument system of [35]. Choose a hash function $H_{\mathrm{FS}} : \{0,1\}^* \to \{-\bar{p}, \ldots, \bar{p}\}^{\kappa}$, for some $\kappa = \mathcal{O}(\lambda / \log \bar{p})$, which will be modelled as a random oracle in the security analysis.
5. Choose a full-rank difference function $\mathsf{FRD} : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n \times n}$ such as the one in [1]; two universal hash functions $H_{\mathsf{UH}} : \mathbb{Z}_p^{m_{\mathsf{LTF}}} \to \mathbb{Z}_p^n, H'_{\mathsf{UH}} : \mathbb{Z}_p^{m_{\mathsf{LTF}}} \to \mathbb{Z}_p^n$ keyed by two uniformly random matrices $\mathbf{U}_{\mathsf{UH}}, \mathbf{U}'_{\mathsf{UH}} \hookleftarrow U(\mathbb{Z}_p^{n \times m_{\mathsf{LTF}}})$; and a collision resistant hash function $H_0 : \{0,1\}^* \to \mathbb{Z}_p^n \setminus \{\mathbf{0}^n\}$.
6. Select a digital signature algorithm[7] Σ able to sign any bitstring.

The final output is $\mathsf{par} = (\mathsf{par}_{\mathsf{PRF}}, \mathsf{par}_{sig}, \mathsf{par}_{\mathsf{LTF}}, \mathbf{F}, \mathsf{FRD}, \mathbf{U}_{\mathsf{UH}}, \mathbf{U}'_{\mathsf{UH}}, H_0, ek_{\mathsf{LTF}}, H_{\mathrm{FS}}, CK, \Sigma)$.

BKeyGen$(1^{\lambda}, \mathsf{par})$**:** The bank $\mathcal{B}$ generates a key pair for the signature scheme by conducting the following steps.

1. Sample $(\mathbf{A}, \mathbf{T}_{\mathbf{A}}) \hookleftarrow \mathsf{TrapGen}(1^n, 1^{m_s}, q_s)$ (details are provided in the full version of this work [16]) so that $\mathbf{T}_{\mathbf{A}}$ is a short basis of $\Lambda_{q_s}^{\perp}(\mathbf{A})$ that allows $\mathcal{B}$ to sample Gaussian vectors in $\Lambda_{q_s}^{\perp}(\mathbf{A})$ with parameter σ.
2. Choose uniform $\mathbf{A}_0, \ldots, \mathbf{A}_{\ell} \hookleftarrow U(\mathbb{Z}_{q_s}^{n \times m_s})$.
3. Choose $\mathbf{D} \hookleftarrow U(\mathbb{Z}_{q_s}^{n \times m_s/2})$ and $\mathbf{u} \hookleftarrow U(\mathbb{Z}_{q_s}^n)$.

The key pair consists of $PK_{\mathcal{B}} := \left(\mathbf{A}, \{\mathbf{A}_j\}_{j=0}^{\ell}, \mathbf{D}, \mathbf{u}\right)$ and $SK_{\mathcal{B}} := \mathbf{T}_A$.

UKeyGen $\left(1^{\lambda}, \mathsf{par}\right)$**:** Choose a secret key $SK_{\mathcal{U}} := \mathbf{e}_u \hookleftarrow U(\{0,1\}^m)$ and set the public key to be $PK_{\mathcal{U}} := \mathbf{F} \cdot \mathbf{e}_u \in \mathbb{Z}_p^n$.

MKeyGen$(1^{\lambda}, \mathsf{par})$**:** Generate and output $(SK_{\mathcal{M}}, PK_{\mathcal{M}}) \leftarrow \Sigma.\mathsf{Keygen}(1^{\lambda})$.

Withdraw $\left(\mathcal{U}(PK_{\mathcal{B}}, SK_{\mathcal{U}}, 2^L), \mathcal{B}(PK_{\mathcal{U}}, SK_{\mathcal{B}}, 2^L)\right)$**:** A user $\mathcal{U}$ withdraws a wallet with 2^L coins from a bank $\mathcal{B}$ by engaging in the following protocol:

[7] Any EUF-CMA secure scheme Σ can be selected here.

1. $\mathcal{U}$ picks a PRF key $\mathbf{k} \hookleftarrow U(\mathbb{Z}_q^m)$ and computes its binary decomposition $\tilde{\mathbf{k}} = \mathsf{vdec}_{m,q-1}(\mathbf{k}) \in \{0,1\}^{\bar{m}}$. Then, $\mathcal{U}$ commits to the 2-block message $(\mathbf{e}_u, \tilde{\mathbf{k}}) \in \{0,1\}^m \times \{0,1\}^{\bar{m}}$ by sampling $\mathbf{r}_0 \hookleftarrow D_{\mathbb{Z}^{m_s},\sigma}$ and sending

$$\mathbf{c}_{\mathcal{U}} = \mathbf{D}_0' \cdot \mathbf{r}_0 + \mathbf{D}_1 \cdot \mathbf{e}_u + \mathbf{D}_2 \cdot \tilde{\mathbf{k}} \in \mathbb{Z}_{q_s}^n$$

to $\mathcal{B}$. In addition, $\mathcal{U}$ generates an interactive zero-knowledge argument of knowledge of an opening $(\mathbf{r}_0, \mathbf{e}_u, \tilde{\mathbf{k}})$ such that $PK_{\mathcal{U}} = \mathbf{F} \cdot \mathbf{e}_u$ with $\mathcal{B}$. This argument of knowledge can be instantiated using the methods of [35] by applying the technique of [14] to parallel repetitions.[8]
2. If the argument of $\mathcal{U}$ verifies, then $\mathcal{B}$ extends the commitment $\mathbf{c}_{\mathcal{U}}$ by sampling $\mathbf{r}_1 \hookleftarrow D_{\mathbb{Z}^{m_s},\sigma}$, and computing $\mathbf{c}_{\mathcal{U}}' = \mathbf{c}_{\mathcal{U}} + \mathbf{D}_0'' \cdot \mathbf{r}_1$. Next $\mathcal{B}$ chooses $\tau \hookleftarrow U(\{0,1\}^{\ell})$, defines $\mathbf{u}_{\mathcal{U}} = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q_s-1}(\mathbf{c}_{\mathcal{U}}')$, sets

$$\mathbf{A}_\tau := [\mathbf{A} | \mathbf{A}_0 + \sum_{j=1}^{\ell} \tau[j] \cdot \mathbf{A}_j] \in \mathbb{Z}_{q_s}^{n \times 2m_s}$$

and computes a short basis $\mathbf{T}_\tau$ of $\Lambda_{q_s}^{\perp}(\mathbf{A}_\tau)$ using $\mathbf{T}_A$. Using $\mathbf{T}_\tau$, it then samples a short vector $\mathbf{v} \hookleftarrow D_{\Lambda_{q_s}^{\mathbf{u}_{\mathcal{U}}}(\mathbf{A}_\tau),\sigma}$ and sends $(\tau, \mathbf{v}, \mathbf{r}_1)$ to $\mathcal{U}$.
3. $\mathcal{U}$ verifies that $\|\mathbf{v}\| \leq \sigma\sqrt{2m_s}$, $\|\mathbf{r}_1\| \leq \sigma\sqrt{m_s}$ and

$$\mathbf{A}_\tau \cdot \mathbf{v} = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q_s-1}(\mathbf{c}_{\mathcal{U}} + \mathbf{D}_0'' \cdot \mathbf{r}_1) \in \mathbb{Z}_{q_s}^n.$$

If so, $\mathcal{U}$ sets $\mathbf{r} = (\mathbf{r}_0^\top \mid \mathbf{r}_1^\top)^\top \in \mathbb{Z}_{q_s}^{2m_s}$ and stores the wallet $\mathcal{W} := \big(\mathbf{e}_u, \mathbf{k}, \mathsf{Sig}_{\mathcal{B}} = (\tau, \mathbf{v}, \mathbf{r}), J = 0\big)$ whereas $\mathcal{B}$ records a debit of 2^L for the account associated to $PK_{\mathcal{U}}$.

Spend $(\mathcal{U}(\mathcal{W}, PK_{\mathcal{B}}, PK_{\mathcal{M}}), \mathcal{M}(SK_{\mathcal{M}}, PK_{\mathcal{B}}, \mathsf{info}))$**:** A user $\mathcal{U}$ in possession of a wallet $\mathcal{W} = \big(\mathbf{e}_u, \mathbf{k}, \mathsf{Sig}_{\mathcal{B}} = (\tau, \mathbf{v}, \mathbf{r}), J\big)$ wants to spend a coin with $\mathcal{M}$. If $J > 2^L - 1$, $\mathcal{U}$ outputs $\perp$. Otherwise, they run the following protocol:

1. $\mathcal{U}$ generates a digital coin by first hashing the transaction information to $R = H_0(PK_{\mathcal{M}}, \mathsf{info}) \in \mathbb{Z}_p^n$ before conducting the following steps.
   a. Compute a BLMR PRF evaluation on the standard binary representation of J in $\{0,1\}^L$ using key $\mathbf{k} \in \mathbb{Z}_q^m$; i.e., set

$$\mathbf{y}_{\mathbf{k}} = \left\lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-j]} \cdot \mathbf{k} \right\rfloor_p$$

   and let $\tilde{\mathbf{y}}_{\mathbf{k}} = \mathsf{bin}_p(\mathbf{y}_{\mathbf{k}}) \in \{0,1\}^{m_{\mathsf{LTF}}}$ its standard bit-decomposition.
   b. Using ek_{LTF}, compute $\mathbf{y}_1 = F_{\mathsf{LTF}}(\tilde{\mathbf{y}}_{\mathbf{k}})$ and $\mathbf{y}_2 = PK_{\mathcal{U}} + H_{\mathsf{UH}}(\tilde{\mathbf{y}}_{\mathbf{k}})$ to form the serial number $\mathbf{y}_S := (\mathbf{y}_1, \mathbf{y}_2) \in \mathbb{Z}_{q_{\mathsf{LTF}}}^{n_{\mathsf{LTF}}+\bar{n}_{\mathsf{LTF}}} \times \mathbb{Z}_p^n$.
   c. Compute the security tag $\mathbf{y}_T = PK_{\mathcal{U}} + \mathsf{FRD}(R) \cdot H_{\mathsf{UH}}'(\tilde{\mathbf{y}}_{\mathbf{k}}) \in \mathbb{Z}_p^n$.
   d. Generate a non-interactive argument of knowledge π_K to show knowledge of $(J, \mathbf{k}, \mathbf{e}_u, (\tau, \mathbf{v}, \mathbf{r}))$ such that:

[8] Technically, we should add a CRS to par but we leave this implicit for simplicity

- The vector $\mathbf{k}$ and secret key $\mathbf{e}_u$ associated with $\mathcal{W}$ and $PK_{\mathcal{U}}$ have been certified by $\mathcal{B}$ through the signature $(\tau, \mathbf{v}, \mathbf{r})$.
- $\mathbf{y}_S$ and $\mathbf{y}_T$ were computed correctly using par, the secret key $\mathbf{e}_u$, the PRF seed $\mathbf{k}$ and a valid $J \in \{0, \ldots, 2^{L-1}\}$.

More precisely, letting $\mathbf{y}_S = (\mathbf{y}_1, \mathbf{y}_2)$, π_K argues knowledge of $(J, \mathbf{k}, \mathbf{e}_u, (\tau, \mathbf{v}, \mathbf{r}))$ where $J \in \{0,1\}^L$, $\mathbf{k} \in \mathbb{Z}_q^m$, $\mathbf{e}_u \in \{0,1\}^m$, $\tau \in \{0,1\}^\ell$, $\mathbf{v} \in \mathbb{Z}^{2m_s}$ s.t. $\|\mathbf{v}\|_\infty \leq \sigma\sqrt{2m_s}$ and $\mathbf{r} \in \mathbb{Z}^{m_s}$ s.t. $\|\mathbf{r}\|_\infty \leq \sigma\sqrt{2m_s}$, satisfying the relations

$$
\begin{aligned}
&[\mathbf{A} \mid \mathbf{A}_0 + \sum_{j=1}^{\ell} \tau[j] \cdot \mathbf{A}_j] \cdot \mathbf{v} \\
&\quad = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n, q_s - 1}\left([\mathbf{D}_0' | \mathbf{D}_0''] \cdot \mathbf{r} + \mathbf{D}_1 \cdot \mathbf{e}_u + \mathbf{D}_2 \cdot \mathsf{vdec}_{m, q-1}(\mathbf{k}) \right) \\
\mathbf{y}_1 &= F_{\mathsf{LTF}}\left(\mathsf{bin}_p\left(\left\lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-j]} \cdot \mathbf{k} \right\rfloor_p \right) \right) \in \mathbb{Z}_{q_{\mathsf{LTF}}}^{n_{\mathsf{LTF}} + \bar{n}_{\mathsf{LTF}}} \\
\mathbf{y}_2 &= \mathbf{F} \cdot \mathbf{e}_u + H_{\mathsf{UH}}\left(\mathsf{bin}_p\left(\left\lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-j]} \cdot \mathbf{k} \right\rfloor_p \right) \right) \in \mathbb{Z}_p^n
\end{aligned}
$$

$$
\mathbf{y}_T = \mathbf{F} \cdot \mathbf{e}_u + \mathsf{FRD}(R) \cdot H'_{\mathsf{UH}}\left(\mathsf{bin}_p\left(\left\lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-j]} \cdot \mathbf{k} \right\rfloor_p \right) \right) \in \mathbb{Z}_p^n
$$

The non-interactive argument π_K is produced by running the proof described in Section 5.2 $\kappa = \mathcal{O}(\lambda / \log \bar{p})$ times in parallel and using the Fiat-Shamir heuristic with random oracle H_{FS}. We may write

$$
\pi_K = \left(\{\mathsf{Comm}_{K,j}\}_{j=1}^{\kappa}, \mathsf{Chall}_K, \{\mathsf{Resp}_{K,j}\}_{j=1}^{\kappa} \right)
$$

where $\mathsf{Chall}_K = H_{\mathrm{FS}}(\mathsf{par}, R, \mathbf{y}_S, \mathbf{y}_T, \{\mathsf{Comm}_{K,j}\}_{j=1}^{\kappa})$.

$\mathcal{U}$ sends $\mathsf{coin} = (\mathsf{info}', PK_{\mathcal{M}}, \mathbf{y}_S, \mathbf{y}_T, \pi_K)$ to $\mathcal{M}$.

2. If $\mathsf{info}' = \mathsf{info}$ and $\mathsf{VerifyCoin}(\mathsf{par}, PK_{\mathcal{B}}, \mathsf{coin})$ outputs 1, then $\mathcal{M}$ outputs coin. Otherwise, $\mathcal{M}$ outputs $\perp$. In either case, $\mathcal{U}$ outputs an updated wallet $\mathcal{W}'$ where J is increased by 1.

VerifyCoin $(PK_{\mathcal{B}}, \mathsf{coin})$**:** Parse the coin as $\mathsf{coin} = (\mathsf{info}, PK_{\mathcal{M}}, \mathbf{y}_S, \mathbf{y}_T, \pi_K)$ and output 1 if and only if π_K verifies.

VerifyDeposit $(PK_{\mathcal{B}}, PK_{\mathcal{M}}, \mathsf{coin}, \mu)$**:** If $\mathsf{VerifyCoin}(PK_{\mathcal{B}}, \mathsf{coin}) = 0$, return 0. Otherwise, return 1 if and only if μ is a valid signature on coin with respect to $PK_{\mathcal{M}}$: i.e., $\Sigma.\mathsf{Verify}(PK_{\mathcal{M}}, \mu, \mathsf{coin}) = 1$.

Deposit $(\mathcal{M}(SK_{\mathcal{M}}, \mathsf{coin}, PK_{\mathcal{B}}), \mathcal{B}(PK_{\mathcal{M}}, SK_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}}))$**:** $\mathcal{M}$ and $\mathcal{B}$ interact in the following way.

1. $\mathcal{M}$ sends $\mathsf{coin} = (\mathsf{info}, PK_{\mathcal{M}}, \mathbf{y}_S, \mathbf{y}_T, \pi_K)$ to $\mathcal{B}$ along with a signature $\mu = \Sigma.\mathsf{Sign}(SK_{\mathcal{M}}, \mathsf{coin})$.
2. If $\mathsf{VerifyDeposit}\,(PK_{\mathcal{B}}, PK_{\mathcal{M}}, \mathsf{coin}, \mu) = 0$ or $\mathsf{VerifyCoin}\,(PK_{\mathcal{B}}, \mathsf{coin}) = 0$, then $\mathcal{B}$ outputs $\perp$. If info and $PK_{\mathcal{M}}$ are found in $\mathsf{state}_{\mathcal{B}}$, then $\mathcal{B}$ outputs the corresponding entry $(\mathsf{coin}', PK_{\mathcal{M}}, \mu', \mathbf{y}'_S)$. If the serial number $\mathbf{y}_S$ contained in coin is not found in $\mathsf{state}_{\mathcal{B}}$, then $\mathcal{B}$ accepts the coin, adds the tuple $(\mathsf{coin}, PK_{\mathcal{M}}, \mu, \mathbf{y}_S)$ to $\mathsf{state}_{\mathcal{B}}$ and credits $\mathcal{M}$'s account. If there exists a tuple $(\mathsf{coin}, PK'_{\mathcal{M}}, \mu', \mathbf{y}_S)$ in $\mathsf{state}_{\mathcal{B}}$, then $\mathcal{B}$ outputs such a tuple.

Identify $(PK_{\mathcal{B}}, \mathsf{coin}_1, \mathsf{coin}_2)$**:** Parse $\mathsf{coin}_i = (\mathsf{info}_i, PK_{\mathcal{M}_i}, \mathbf{y}_{S,i}, \mathbf{y}_{T,i}, \pi_{K,i})$ for each $i \in \{1, 2\}$. If any of the following conditions hold, output $\perp$:

- $\mathbf{y}_{S,1} \neq \mathbf{y}_{S,2}$,
- $\mathsf{VerifyCoin}\,(\mathsf{par}, PK_{\mathcal{B}}, \mathsf{coin}_1)$ or $\mathsf{VerifyCoin}\,(\mathsf{par}, PK_{\mathcal{B}}, \mathsf{coin}_2) \neq 1$,
- $(\mathsf{info}_1, PK_{\mathcal{M}_1}) = (\mathsf{info}_2, PK_{\mathcal{M}_2})$.

Otherwise, compute $\mathbf{y}'_T = (\mathsf{FRD}(R_1) - \mathsf{FRD}(R_2))^{-1} \cdot (\mathbf{y}_{T,1} - \mathbf{y}_{T,2}) \in \mathbb{Z}_p^n$ with $R_i = H_0(PK_{\mathcal{M}_i}, \mathsf{info}_i)$ and set $PK_{\mathcal{U}} = \mathbf{y}_{T,1} - \mathsf{FRD}(R_1) \cdot \mathbf{y}'_T \in \mathbb{Z}_p^n$. Note that this calculation is performed using publicly known values, so the proof of guilt of a double spender is simply $\Pi_G = (\mathsf{coin}_1, \mathsf{coin}_2)$. The output of this algorithm is then the pair $(PK_{\mathcal{U}}, \Pi_G)$.

5 Zero-Knowledge Arguments with Soundness Error $1/\mathbf{poly}(\lambda)$ in Standard Lattices

We proceed in two steps to describe the ZKAoK used to spend a coin. We first describe an argument of knowledge of a (seed,input) pair generating a given BLMR evaluation. We then extend this to capture the whole statement proved by a user during a spending. For the ZKAoK in the withdrawal protocol, we directly rely on results of [35]. Throughout our construction, we use the argument system of Yang *et al.* [35] which was originally proved computationally honest-verifier ZK (HVZK) with polynomial soundness error. However, we can use known techniques to transform parallel repetitions of this protocol into a 3-round, malicious verifier ZK protocol with negligible soundness error in the CRS model [14]. This is how we instantiate the *interactive* ZKAoK in the withdrawal protocol. In the spend protocol, we use the standard result that the Fiat-Shamir transform [21] applied to parallel repetitions of an HVZK protocol yields a NIZK argument in the ROM. We also note that one may use a statistically hiding configuration of the commitment scheme from [5] instead of the more efficient computationally hiding configuration chosen in [35] to obtain *statistical* ZK arguments.

5.1 Zero-Knowledge Arguments for the BLMR PRF

We extend the protocol of Yang *et al.* [35] to build a ZKAoK of a (seed,input) pair producing a given BLMR evaluation. A similar result for the GGM-based

PRF implied by [4] is provided in the full version [16], leading to a more efficient instantiation.

In [35], Yang *et al.* provide an argument of knowledge for the "instance-relation" set given by

$$\mathcal{R}^* = \left\{ \left((\mathbf{M}', \mathbf{y}', \mathcal{M}), \mathbf{x}' \right) : \begin{smallmatrix} \mathbf{M}' \cdot \mathbf{x}' = \mathbf{y}' \bmod q \ \wedge \\ \forall (h,i,j) \in \mathcal{M}, \mathbf{x}'[h] = \mathbf{x}'[i] \cdot \mathbf{x}'[j] \bmod q \end{smallmatrix} \right\}. \tag{1}$$

where $\mathbf{M}' \in \mathbb{Z}_q^{m' \times n'}, \mathbf{y}' \in \mathbb{Z}_q^{m'}$ and $\mathcal{M} \subseteq [n'] \times [n'] \times [n']$, for some prime power q. The tuple $(\mathbf{M}', \mathbf{y}', \mathcal{M})$ is the instance whereas $\mathbf{x}' \in \mathbb{Z}_q^{n'}$ is the witness. By carefully crafting each of these elements, we show that a proof of correct evaluation of the BLMR PRF is an instance of this argument of knowledge.

Indeed, recall that, for any seed $\mathbf{k}$ and input $x \in \{0,1\}^L$, the PRF output is defined as $\mathbf{y} = \left\lfloor \prod_{i=1}^{L} \mathbf{P}_{x_{L+1-i}} \cdot \mathbf{k} \right\rfloor_p$, where $\mathbf{P}_0, \mathbf{P}_1 \in \{0,1\}^{m \times m}$ are public parameters and p is a prime power dividing q. If we write $\mathbf{y}_j = \prod_{i=L+1-j}^{L} \mathbf{P}_{x_{L+1-i}} \cdot \mathbf{k}$ for $j \in [L]$, we can represent a PRF evaluation using the linear system over $\mathbb{Z}_q$:

$$\begin{aligned} \mathbf{y}_1 - \mathbf{P}_0 \cdot (1 - x_1)\mathbf{k} - \mathbf{P}_1 \cdot x_1 \mathbf{k} &= \mathbf{0} \\ \mathbf{y}_2 - \mathbf{P}_0 \cdot (1 - x_2)\mathbf{y}_1 - \mathbf{P}_1 \cdot x_2 \mathbf{y}_1 &= \mathbf{0} \\ &\vdots \\ \mathbf{y}_L - \mathbf{P}_0 \cdot (1 - x_L)\mathbf{y}_{L-1} - \mathbf{P}_1 \cdot x_L \mathbf{y}_{L-1} &= \mathbf{0} \\ \mathbf{y}_L - \mathbf{e} &= \frac{q}{p} \cdot \mathbf{y} \end{aligned}$$

where $\mathbf{e} \in [0, q/p]^m$. This system is a linear system in the (quadratic) unknowns $(1-x_1)\mathbf{k}, x_1\mathbf{k}, (1-x_2)\mathbf{y}_1, x_2\mathbf{y}_1, \ldots, (1-x_L)\mathbf{y}_{L-1}, x_L\mathbf{y}_{L-1}, \mathbf{y}_L, \mathbf{e}$. As a first step towards transforming our system into one captured by $\mathcal{R}^*$, we can embed the above system in a larger system whose solution is given by

$$(\mathbf{x}')^\top = \left((\mathbf{x}'_1)^\top, (\mathbf{x}'_2)^\top, (\mathbf{x}'_3)^\top, \tilde{\mathbf{e}}^\top \right) \tag{2}$$

where

- $(\mathbf{x}'_1)^\top = \big((1-x_1), x_1, \ldots, (1-x_L), x_L \big) \in \{0,1\}^{2L}$,
- $(\mathbf{x}'_2)^\top = \big(\mathbf{y}_0^\top, \mathbf{y}_1^\top, \ldots, \mathbf{y}_L^\top \big) \in \mathbb{Z}_q^{(L+1)\cdot m}$, with $\mathbf{y}_0 := \mathbf{k}$,
- $\mathbf{x}'_3 \in \mathbb{Z}_q^{2L \cdot m}$ is of the form

$$(\mathbf{x}'_3)^\top = \big((1-x_1)\mathbf{y}_0, x_1\mathbf{y}_0, (1-x_2)\mathbf{y}_1, x_2\mathbf{y}_1, \ldots, (1-x_L)\mathbf{y}_{L-1}, x_L\mathbf{y}_{L-1} \big),$$

- $\tilde{\mathbf{e}} = \mathsf{vdec}_{m, \frac{q}{p}-1}(\mathbf{e}) \in \{0,1\}^{m \cdot \left(\lfloor \log(\frac{q}{p}-1) \rfloor + 1 \right)}$, which ensures that $\|\mathbf{e}\|_\infty < q/p$.

One aspect of this extended solution is that every consecutive pair of entries of $\mathbf{x}'_1$ is either $(0,1)$ or $(1,0)$. In other words, each consecutive pair of entries of $\mathbf{x}'_1$ sums to 1 and is binary. The fact that consecutive pairs add to 1 can be captured by a linear constraint that will constitute the first block of our matrix

$\mathbf{M}'$. Next, the fact that the entries of $\mathbf{x}'_1$ are binary may be captured by the set of equations $\mathbf{x}'_1[i] = \mathbf{x}'_1[i] \cdot \mathbf{x}'_1[i]$. In fact, proving this relation only for even i is sufficient as $\mathbf{x}'_1[2i] \in \{0,1\}$ and $\mathbf{x}'_1[2i] + \mathbf{x}'_1[2i-1] = 1$ implies $\mathbf{x}'_1[2i-1] \in \{0,1\}$.

The next part of a valid solution's structure is that entries of $\mathbf{x}'_3$ are the result of multiplying entries of $\mathbf{x}'_1$ and $\mathbf{x}'_2$. This can be written as $\mathbf{x}'_3[h'] = \mathbf{x}'_1[i'] \cdot \mathbf{x}'_2[j']$ for appropriate choices of h', i', j'. It then only remains to prove that the entries of $\tilde{\mathbf{e}}$ are binary, which is captured by the equations $\tilde{\mathbf{e}}[i] = \tilde{\mathbf{e}}[i] \cdot \tilde{\mathbf{e}}[i]$.

Following the details outlined above, we may represent a BLMR evaluation as the system $\mathbf{M}' \cdot \mathbf{x}' = \mathbf{y}' \bmod q$ for

- $\mathbf{x}' \in \mathbb{Z}_q^{2L+(L+1)\cdot m+2L\cdot m+(\lfloor \log(q/p-1)\rfloor+1)\cdot m}$ which is subject to the following constraints, when parsed as in Equation 2:
  - for $i \in [L]$: $\mathbf{x}'_1[2i] = \mathbf{x}'_1[2i] \cdot \mathbf{x}'_1[2i]$
  - for $(i,j) \in [m] \times [L]$: $\mathbf{x}'_3[2m(j-1)+i] = \mathbf{x}'_1[2j-1] \cdot \mathbf{x}'_2[m(j-1)+i]$ and $\mathbf{x}'_3[2m(j-1)+m+i] = \mathbf{x}'_1[2j] \cdot \mathbf{x}'_2[m(j-1)+i]$
  - for $i \in [(\lfloor \log(q/p-1)\rfloor+1)\cdot m]$: $\tilde{\mathbf{e}}[i] = \tilde{\mathbf{e}}[i] \cdot \tilde{\mathbf{e}}[i]$
- $(\mathbf{y}')^\top = (\overbrace{1,\ldots,1}^{L}, \overbrace{0,\ldots,\ldots,0}^{m\cdot L}, (q/p)\mathbf{y}^\top)$
- $$\mathbf{M}' = \left[\begin{array}{c|c|c|c} \mathbf{I}_L \otimes (1,1) & & & \\ \hline & 0^{mL\times m} \| \mathbf{I}_{m\cdot L} & -\mathbf{I}_L \otimes [\mathbf{P}_0 \| \mathbf{P}_1] & \\ \hline & 0^{m\times L\cdot m} \| \mathbf{I}_m & & -H_{m,q/p-1} \end{array}\right] \tag{3}$$
  where all blank blocks consist of 0 entries.

5.2 Zero-Knowledge Arguments for the Spend Protocol

The previous protocol enables to prove correct evaluation of the BLMR PRF but is not sufficient to produce the proof π_K expected by the merchant during the Spend protocol. In particular, we also need to prove

- knowledge of (short) solutions to linear systems (e.g., the user's secret key);
- knowledge of solutions to an equation involving a subset sum of known-matrix and secret vector multiplications (i.e. the computation of $\mathbf{A}_\tau$);
- correct evaluation of the lossy trapdoor function F_{LTF}.

All these statements can be captured by the relation $\mathcal{R}^*$ from [35], as explained below. Together with our proof of correct PRF evaluation, this means that π_K can be instantiated using only the Yang *et al.* framework. We can then achieve inverse-polynomial soundness error $1/\bar{p}$ in one ZKAoK protocol run. To achieve a soundness error of $2^{-\lambda}$, we only need $\mathcal{O}(\lambda/\log \bar{p})$ repetitions. This clearly improves upon the Stern-type protocols used in [26], which require $\mathcal{O}(\lambda)$ repetitions.

Remark 1. It should be noted that we have different equations over various moduli in our Spend protocol. However, as long as q is a prime power and all remaining moduli divide q, we may lift all equations to use the modulus q. For example, to lift an equation over $\mathbb{Z}_{q'}$ to an equation over $\mathbb{Z}_q$ where q' divides q, we simply multiply by $q/q' \in \mathbb{Z}$. We will use this trick in what follows.

The explicit linear system. Transforming the argument of knowledge produced by a user during the Spend protocol into an instance of the Yang *et al.* protocol is far from trivial as there are several details to address. Besides the moduli issue mentioned above, we indeed need to juggle with two different types of binary decomposition in order to ensure security of the whole system.

We use the notation from the Spend protocol specification in Section 4. We further parse $\mathbf{v}$ as $(\mathbf{v}_1, \mathbf{v}_2)$, where $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{Z}^{m_s}$. Also, we define $\sigma' := \lfloor \sigma\sqrt{m_s}+1 \rfloor$ and $\mathbf{v}_i^+ = \mathbf{v}_i + \sigma' \cdot \mathbf{1}$ for $i \in \{1,2\}$, where $\mathbf{1}$ denotes the all-one vector. This implies that valid values of $\mathbf{v}_i$ (i.e., such that $\|\mathbf{v}_i\|_\infty \leq \sigma'$) give rise to $\mathbf{v}_i^+ \in [0, 2\sigma']^{m_s}$. We also set $\mathbf{r}^+ := \mathbf{r} + \sqrt{2}\sigma' \cdot \mathbf{1}$ so that $\mathbf{r}^+ \in [0, 2\sqrt{2}\sigma']^{2m_s}$ for valid choices of $\mathbf{r}$ (i.e. values such that $\|\mathbf{r}\|_\infty \leq \sqrt{2}\sigma'$). We can then define $\tilde{\mathbf{v}}_i := \mathsf{vdec}_{m_s,2\sigma'}\left(\mathbf{v}_i^+\right)$ for $i \in \{1,2\}$, $\tilde{\mathbf{r}} := \mathsf{vdec}_{2m_s,2\sqrt{2}\sigma'}\left(\mathbf{r}^+\right)$, $\tilde{\mathbf{k}} := \mathsf{vdec}_{m,q-1}\left(\mathbf{k}\right)$ and

$$\widetilde{\mathbf{w}} := \mathsf{vdec}_{n,q_s-1}\left([\mathbf{D}_0' | \mathbf{D}_0''] \cdot \mathbf{r} + \mathbf{D}_1 \cdot \mathbf{e}_u + \mathbf{D}_2 \cdot \tilde{\mathbf{k}} \right).$$

We begin by considering the equation associated to the signature. We can express it as the following linear system over $\mathbb{Z}_q$

$$\begin{aligned}
&\frac{q}{q_s}\Bigg[\mathbf{A}\left(\mathbf{H}_{m_s,2\sigma'} \cdot \tilde{\mathbf{v}}_1 - \sigma'\mathbf{1}\right) + \\
&\qquad \mathbf{A}_0\left(\mathbf{H}_{m_s,2\sigma'} \cdot \tilde{\mathbf{v}}_2 - \sigma'\mathbf{1}\right) + \\
&\qquad\qquad \sum_{j=1}^{\ell} \mathbf{A}_j\left(\mathbf{H}_{m_s,2\sigma'} \cdot (\tau[j] \cdot \tilde{\mathbf{v}}_2) - \sigma'\tau[j] \cdot \mathbf{1}\right) - \mathbf{D} \cdot \tilde{\mathbf{w}}\Bigg] = \frac{q}{q_s}\mathbf{u} \\
&\frac{q}{q_s}\Bigg[\mathbf{H}_{n,q_s-1} \cdot \tilde{\mathbf{w}} - \Bigg([\mathbf{D}_0' | \mathbf{D}_0''] \left(\mathbf{H}_{2m_s,2\sqrt{2}\sigma'} \cdot \tilde{\mathbf{r}} - \sqrt{2}\sigma'\mathbf{1}\right) + \\
&\qquad\qquad \mathbf{D}_1 \cdot \mathbf{e}_u + \mathbf{D}_2 \cdot \tilde{\mathbf{k}}\Bigg)\Bigg] = \mathbf{0} \\
&\qquad\qquad \mathbf{H}_{m,q-1} \cdot \tilde{\mathbf{k}} - \mathbf{k} = \mathbf{0},
\end{aligned}$$

whose solution is $\mathbf{x}_1 := \left(\tau, \tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tau[1] \cdot \tilde{\mathbf{v}}_2, \ldots, \tau[\ell] \cdot \tilde{\mathbf{v}}_2, \tilde{\mathbf{w}}, \tilde{\mathbf{r}}, \mathbf{e}_u, \mathbf{k}, \tilde{\mathbf{k}}\right)$, with some quadratic constraints amongst unknowns.

We next consider the evaluation of $\mathbf{y}_1$, as written in the Spend protocol. Here a subtlety arises as we need to use two different types of binary decomposition. So far, we have only used the $\mathsf{vdec}_{m,p-1}$ function because it allows achieving exact soundness with the proofs of Yang *et al.* Unfortunately, the decomposition of an integer according to the sequences $B_1, \ldots, B_{\delta_{p-1}}$ implicitly defined by $\mathsf{vdec}_{m,p-1}$ (see Section 2) may not be unique, which might lead to undetected frauds in our system. We will then also use the *standard* binary decomposition (that is unique) to ensure that the user is not evaluating F_{LTF} on two different decompositions of the same PRF output. It then remains to prove consistency of both decompositions, which is explained below.

Concretely, let $\tilde{\mathbf{y}}_{\mathbf{k}}$ denote the standard binary decomposition of the PRF output $\mathbf{y}_{\mathbf{k}} = \left\lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-j]} \cdot \mathbf{k} \right\rfloor_p$. Importantly, we must ensure that $\tilde{\mathbf{y}}_{\mathbf{k}}$ does

really correspond to binary decomposition of a vector in $[0, p-1]^m$ rather than some larger space. Alternatively, we need to ensure that $\mathbf{y_k}$ (which is unknown) has entries in $[0, p-1]$. We achieve this by considering $\tilde{\mathbf{y}}'_{\mathbf{k}} = \mathsf{vdec}_{m,p-1}(\mathbf{y_k})$. By multiplying the evaluation equation of $\mathbf{y}_1$ by q/q_{LTF} and denoting the LTF key ek_{LTF} as $\mathbf{B}_{\mathsf{LTF}} \in \mathbb{Z}_{q_{\mathsf{LTF}}}^{(n_{\mathsf{LTF}}+\bar{n}_{\mathsf{LTF}})\times m_{\mathsf{LTF}}}$, we can derive the following equations over $\mathbb{Z}_q$:

$$\boxed{\begin{aligned} \frac{q}{q_{\mathsf{LTF}}}\mathbf{B}_{\mathsf{LTF}} \cdot \tilde{\mathbf{y}}_{\mathbf{k}} &= \frac{q}{q_{\mathsf{LTF}}} \cdot \mathbf{y}_1 \\ \mathbf{y}_k - \mathbf{H}_{m,p-1} \cdot \tilde{\mathbf{y}}'_{\mathbf{k}} &= \mathbf{0} \\ \mathbf{y}_k - \mathbf{I}_m \otimes \left(1, 2, \ldots, 2^{\lceil \log p \rceil}\right) \cdot \tilde{\mathbf{y}}_{\mathbf{k}} &= \mathbf{0} \end{aligned}}$$

Conveniently, the restriction that the entries of $\tilde{\mathbf{y}}_{\mathbf{k}}$ and $\tilde{\mathbf{y}}'_{\mathbf{k}}$ are binary is easily captured using quadratic constraints. Therefore all boxed equations so far constitute a linear system whose solution is $\mathbf{x_2} := (\mathbf{x}_1 \| \tilde{\mathbf{y}}_{\mathbf{k}}, \tilde{\mathbf{y}}'_{\mathbf{k}}, \mathbf{y_k})$, subject to some quadratic constraints that can easily be handled with the Yang *et al.* framework. However, we still need some equations to ensure that $\mathbf{y_k}$ is computed correctly as a BLMR PRF output. In order to describe these equations, we will use the observations from Section 5.1 and the matrix $\mathbf{M}'$ given in Equation (3). In particular, we set the unknown vector

$$\begin{aligned} \mathbf{x_k} =& (1 - J[1], J[1], \ldots, 1 - J[L], J[L], \mathbf{yk}_0, \ldots, \mathbf{yk}_L, \\ & (1 - J[1])\mathbf{yk}_0, J[1]\mathbf{yk}_0, \ldots, (1 - J[L])\mathbf{yk}_{L-1}, J[L]\mathbf{yk}_{L-1}, \mathbf{e_k}) \end{aligned}$$

where $\mathbf{yk}_i \in \mathbb{Z}_q^m$ for $i \in [0, L]$ and $\mathbf{e_k} \in \{0,1\}^{m \cdot \left(\lfloor \log(\frac{q}{p}-1) \rfloor + 1\right)}$. As noted in Section 5.1 (and shown by the form of $\mathbf{x_k}$), the constraints on these unknown vectors are quadratic as required. To capture the PRF computation, we extend the vector of unknowns by defining $\mathbf{x}_3 := (\mathbf{x}_2 \| \mathbf{x_k})$. We then add the following to the boxed linear equations over $\mathbb{Z}_q$ above (where $\mathbf{M}'$ is defined in Equation (3)):

$$\boxed{\begin{aligned} \mathbf{yk}_0 - \mathbf{k} &= \mathbf{0} \\ \mathbf{M}' \cdot \mathbf{x_k} - \left(0^{(m+1)\cdot L}, \frac{q}{p}\mathbf{y}_{\mathbf{k}}^\top\right)^\top &= (1^L, 0^{m\cdot(L+1)})^\top \end{aligned}}$$

Finally, it remains to prove that $\mathbf{y}_2$ and $\mathbf{y}_T$ are well-formed. This consists in proving the following relation over $\mathbb{Z}_q$:

$$\boxed{\begin{aligned} \frac{q}{p}\mathbf{F} \cdot \mathbf{e}_u + \frac{q}{p}\mathbf{U}_{\mathsf{UH}} \cdot \tilde{\mathbf{y}}_{\mathbf{k}} &= \frac{q}{p}\mathbf{y}_2 \\ \frac{q}{p}\mathbf{F} \cdot \mathbf{e}_u + \frac{q}{p}\mathsf{FRD}(R) \cdot \mathbf{U}'_{\mathsf{UH}} \cdot \tilde{\mathbf{y}}_{\mathbf{k}} &= \frac{q}{p}\mathbf{y}_T, \end{aligned}}$$

where the witnesses are already included in $\mathbf{x}_3$.

We have shown that the whole statement proved during the Spend protocol can be expressed as the collection of the boxed linear systems with a vector $\mathbf{x}_3$ of unknowns subject to quadratic constraints supported by the protocol of [35].

6 Security Proofs

In this section and the full version [16], we prove Theorem 1, which states that our construction provides all the required security properties.

Theorem 1. *Our construction is a secure e-cash system in the random oracle model assuming that the following conditions hold:*

- *The* $\mathsf{SIS}_{n,m_s,q_s,\beta'}$ *for* $\beta' = \mathcal{O}\left(\sigma^2 m_s^{1/2}(m_s + m\log q)\right)$ *and* $\mathsf{SIS}_{n,m,p,2\sqrt{m}}$ *problems are hard;*
- *Parameters are chosen so that the interactive AoK* Π_1 *in the withdrawal protocol is zero-knolwedge (ZK) and that the non-interactive AoK* Π_2 *in the spend protocol is honest-verifier zero-knowledge (HVZK);*
- *Parameters* m, n, q, p *are chosen so that the BLMR PRF is pseudo-random;*
- *The* $\mathsf{LWE}_{n_{\mathsf{LTF}},m_{\mathsf{LTF}},q_{\mathsf{LTF}},\alpha}$ *problem is hard;*
- Σ *is an EUF-CMA secure signature scheme.*

Proof of Exculpability. Suppose the lossy trapdoor function is sampled in its injective mode. The proof of exculpability relies on the fact that an adversary producing two valid coins with the same serial number must produce at least one fresh proof of knowledge of a secret key underlying an honestly produced public key. In particular, our construction guarantees that this public key is the one that $\mathsf{Identify}$ points to. The ability to produce fresh arguments of knowledge for an honest public key can be used to solve the SIS problem. We first present a lemma about collision probabilities on PRFs with randomly sampled seeds and polynomial-size domain.

Lemma 4. *Let* $\mathsf{PRF} = \left\{\mathsf{PRF}_{\mathbf{k}} : \{0,1\}^L \to \{0,1\}^M \mid \mathbf{k} \in \mathcal{K}\right\}$ *be a family of pseudo-random functions where* $2^L = \mathsf{poly}(\lambda)$ *and* $M = \mathsf{poly}(\lambda)$*. Take any* $N = \mathsf{poly}(\lambda)$ *and sample* $\mathbf{k}_1, \ldots, \mathbf{k}_N \hookleftarrow U(\mathcal{K})$*. The probability that* $\exists (i, j, x_1, x_2) \in [N]^2 \times \{0,1\}^L \times \{0,1\}^L$ *such that* $\mathsf{PRF}_{\mathbf{k}_i}(x_1) = \mathsf{PRF}_{\mathbf{k}_j}(x_2)$ *is negligible.*

Proof. We first describe a challenger algorithm $\mathcal{C}$. In the first stage, $\mathcal{C}$ samples $\mathbf{k}_1, \ldots, \mathbf{k}_N \hookleftarrow U(\mathcal{K})$, samples N uniform functions $U_1, \ldots, U_N : \{0,1\}^L \to \{0,1\}^M$ and samples a challenge bit $b \hookleftarrow U(\{0,1\})$. In the second phase, $\mathcal{C}$ waits for queries $x \in \{0,1\}^L$. If $b = 1$, it answers with $(\mathsf{PRF}_{\mathbf{k}_1}(x), \ldots, \mathsf{PRF}_{\mathbf{k}_N}(x))$. On the other hand, if $b = 0$, it responds with $(U_1(x), \ldots, U_N(x))$. By a standard hybrid argument, no PPT adversary $\mathcal{A}$ can guess the bit b with non-negligible advantage under the assumption that PRF is a PRF family and $N = \mathsf{poly}(\lambda)$. Consider the following adversary $\mathcal{A}^*$ that queries $\mathcal{C}$ on the entire set $\{0,1\}^L$. Denote the response to query x as $(y_{1,x}, \ldots, y_{N,x})$. Now, $\mathcal{A}^*$ outputs $b^* = 1$ if there exists (i, j, x_1, x_2) such that $y_{i,x_1} = y_{j,x_2}$. Otherwise, $\mathcal{A}^*$ outputs $b^* = 0$. Note that, if $b = 0$, the probability that $\mathcal{A}^*$ outputs $b^* = 1$ is equal to

$$1 - \prod_{k=1}^{2^L N} \left(1 - \frac{(k-1)}{2^M}\right)$$

which is negligible since $2^L N = \mathsf{poly}(\lambda)$ and $2^M = 2^{\mathsf{poly}(\lambda)}$. Therefore, under the assumption that PRF is a PRF family, the probability that $\mathcal{A}^*$ outputs $b^* = 1$ when $b = 1$ is also negligible. □

Lemma 5. *Our construction provides strong exculpability in the random oracle model assuming that: (i) The $\mathsf{SIS}_{n,m,p,2\sqrt{m}}$ problem is hard; (ii) Parameters (m, n, p, q) are chosen so that the BLMR PRF is pseudo-random; (iii) Π_1 and Π_2 are ZK and HVZK, respectively; (iv) The protocols underlying Π_1 and Π_2 are arguments of knowledge.*

Recall that a successful adversary returns coin_1 and coin_2 such that $PK_{\mathcal{U}^*} = \mathsf{Identify}(PK_{\mathcal{B}}, \mathsf{coin}_1, \mathsf{coin}_2)$ for honest user $\mathcal{U}^*$. This implies two things:

- First, the two coins have been generated using the public key $PK_{\mathcal{U}^*}$. Indeed, the fact that the identification procedure succeeds implies that these coins share the same serial number $\mathbf{y}_S := (\mathbf{y}_1, \mathbf{y}_2)$. Since the evaluation key of F_{LTF} was sampled in injective mode, the serial number $\mathbf{y}_S$ uniquely determines the value $PK' = \mathbf{y}_2 - H_{\mathsf{UH}}(F_{\mathsf{LTF}}^{-1}(\mathbf{y}_1))$, which underlies both coin_1 and coin_2. Then, the soundness of Π_2 ensures that
$$\mathbf{y}_{T,1} = PK' + \mathsf{FRD}(R_1) \cdot H'_{\mathsf{UH}}(F_{\mathsf{LTF}}^{-1}(\mathbf{y}_1)),$$
$$\mathbf{y}_{T,2} = PK' + \mathsf{FRD}(R_2) \cdot H'_{\mathsf{UH}}(F_{\mathsf{LTF}}^{-1}(\mathbf{y}_1)),$$
which implies that PK' is the public key $PK_{\mathcal{U}^*}$ pointed to by $\mathsf{Identify}$.
- Second, there exists $d \in \{1, 2\}$ such that $\mathsf{coin}_d = (R_d, \mathbf{y}_{S,d}, \mathbf{y}_{T,d}, \pi_{K,d})$ is *not* the result of a $\mathcal{Q}_{\mathsf{Spend}}$-query w.h.p. To see why, consider the case that coin_1 and coin_2 are both the result of $\mathcal{Q}_{\mathsf{Spend}}$-queries, but do not appear in T_{ds}. This occurs if, when sampling polynomially many seeds, one finds $\mathbf{k}, \mathbf{k}'$ satisfying $\mathsf{PRF}_{\mathbf{k}}(J) = \mathsf{PRF}_{\mathbf{k}'}(J')$ for some $(J, J') \in [0, 2^L - 1]^2$. By Lemma 4, this occurs with negligible probability $\mathsf{negl}_1(\lambda)$.

Proof. Using these two observations, we will prove the strong exculpability of our scheme by defining the following sequence of games. Let ϵ be the probability that $\mathcal{A}$ succeeds against the exculpability of our scheme and let Q_w (resp. Q_s) denote the maximal number of $\mathcal{Q}_{\mathsf{Withdraw}}$ queries (resp. $\mathcal{Q}_{\mathsf{Spend}}$ queries).

Game_0: This is exactly the strong exculpability experiment, as defined in Section 2. The probability ϵ_0 that $\mathcal{A}$ succeeds in this game is then exactly ϵ.

$\mathsf{Game}_{1,0}$: In this game, our reduction $\mathcal{S}$ (acting as a challenger in the strong exculpability experiment) proceeds as in Game_0 except that it defines $\mathbf{F}$ as $\bar{\mathbf{A}} \in \mathbb{Z}_p^{n \times m}$, where $\bar{\mathbf{A}}$ is a uniform matrix provided in a $\mathsf{SIS}_{n,m,p,2\sqrt{m}}$ instance. We denote by $\mathbf{e}_{u^*} \in \{0, 1\}^m$ the secret key generated by $\mathcal{S}$ for the accused user $PK_{\mathcal{U}^*} = \mathbf{F} \cdot \mathbf{e}_{u^*}$. Note that $\mathcal{A}$ is given black-box access to H_{FS} and $\mathcal{S}$ answers queries to H_{FS} by returning uniformly random elements of $\{-\bar{p}, \ldots, \bar{p}\}^\kappa$. In addition, $\mathcal{S}$ initialises empty lists of honest users $\mathcal{HU}$ and double-spent coins $\mathcal{T}_{\mathsf{ds}}$. As $\bar{\mathbf{A}}$ is distributed as $\mathbf{F}$ in the original setup, the probability that $\mathcal{A}$ succeeds in this game is $\epsilon_{1,0} = \epsilon_0$.

$\mathsf{Game}_{1,i}$: For $i \in [1, Q_w]$, this game is defined as $\mathsf{Game}_{1,i-1}$, except that $\mathcal{S}$ now answers the i-th $\mathcal{Q}_{\mathsf{Withdraw}}$-query (if any) by running the simulator of Π_1 to simulate the interactive proof generated by the user at this stage. This is done for every user $PK_{\mathcal{U}}$, and not just $PK_{\mathcal{U}^*}$. Any change of behaviour of $\mathcal{A}$ can thus be straightforwardly used against the zero-knowledge property of Π_1. We therefore have $\epsilon_{1,i-1} - \mathtt{Adv}^{\Pi_1}_{ZK}(\mathcal{A}) \leq \epsilon_{1,i}$ for all $i \in [1, Q_w]$.

Game_{1,Q_w+i}: For $i \in [1, Q_s]$, this game is defined as $\mathsf{Game}_{1,Q_w+i-1}$, except that $\mathcal{S}$ now answers the i-th $\mathcal{Q}_{\mathsf{Spend}}$-query (if any) by running the simulator of Π_2 to simulate the non-interactive argument generated by the spender at this stage. This can be done (using only the user's public key $PK_{\mathcal{U}}$) by applying the standard technique of programming the random oracle H_{FS} on new inputs, which only requires the statistical HVZK property of Π_2. The simulation fails whenever the random oracle H_{FS} needs to be programmed at an input that it was previously queried on. However, this happens with negligible probability at most $\mathtt{Coll}_H := (Q_S + Q_H)/2^{\lambda}$, where Q_H is the total number of queries made by $\mathcal{A}$ to H_{FS} and the denominator 2^{λ} is a lower bound on the domain-size of H_{FS}-inputs. Therefore, we can conclude that $\epsilon_{1,Q_w+i-1} - \mathtt{Adv}^{\Pi_2}_{HVZK}(\mathcal{A}) - \mathtt{Coll}_H \leq \epsilon_{1,Q_w+i}$ for all $i \in [1, Q_s]$.

It is important to note that, in $\mathsf{Game}_{1,Q_w+Q_s}$, the reduction $\mathcal{S}$ only needs $PK_{\mathcal{U}^*}$ and not $\mathbf{e}_{u^*}$ to simulate the game. This concretely means that the adversary's view is independent of the preimage $\mathbf{e}_{u^*}$ of $PK_{\mathcal{U}^*}$ selected by $\mathcal{S}$. Thanks to [27, Lemma 8], we know that this preimage is not unique: i.e., there exists at least one vector $\mathbf{e} \in \{0,1\}^m \setminus \{\mathbf{e}_{u^*}\}$ such that $\bar{\mathbf{A}} \cdot \mathbf{e}_{u^*} = \bar{\mathbf{A}} \cdot \mathbf{e} \bmod p$ with all but negligible probability. This observation will be crucial in what follows.

Game_2: Let Q_H be a polynomial bounding the number of random oracle queries made by $\mathcal{A}$ to H_{FS}. Up until $\mathcal{A}$ terminates, $\mathcal{S}$ answers $\mathcal{A}$'s queries as in the previous games, recording the random oracle queries as $(q_1, q_2, \dots)$ and the corresponding uniformly distributed responses as $(h_1, h_2, \dots)$. Our second observation at the beginning of the proof implies that at least one coin coin_d returned by $\mathcal{A}$ is not the result of a $\mathcal{Q}_{\mathsf{Spend}}$-query with overwhelming probability (if none of the coins were generated as a response to $\mathcal{Q}_{\mathsf{Spend}}$-query, then select a random $d \in \{1, 2\}$. Define

$$\pi_{K,d} := \left(\left\{\mathsf{Comm}_{K,d,j}\right\}_{j=1}^{\kappa}, \mathsf{Chall}_{K,d}, \left\{\mathsf{Resp}_{K,d,j}\right\}_{j=1}^{\kappa}\right),$$
$$\mathsf{Chall}_{K,d} := H_{\mathrm{FS}}\left(\mathsf{par}, R, \mathbf{y}_{S,d}, \mathbf{y}_{T,d}, \left\{\mathsf{Comm}_{K,d,j}\right\}_{j=1}^{\kappa}\right).$$

In this game, $\mathcal{S}$ aborts if the above query was not made to H_{FS}. We note that in such a case the proof $\pi_{K,d}$ would only have been acceptable with probability at most $(2\bar{p}+1)^{-\kappa}$. We then have $\epsilon_{1,Q_w+Q_s} - (2\bar{p}+1)^{-\kappa} \leq \epsilon_2$.

From now on, we know that there exists an index $i^* \in [Q_H]$ such that the i^*-th H_{FS}-query is used to produce $\mathsf{Chall}_{K,d}$ (i.e., $\mathsf{Chall}_{K,d} = h_{i^*}$) and that $\mathcal{A}$ succeeds in Game_2 with probability $\epsilon_2 \geq \epsilon - Q_w \cdot \mathtt{Adv}^{\Pi_1}_{ZK}(\mathcal{A}) - Q_s(\mathtt{Adv}^{\Pi_2}_{HVZK}(\mathcal{A}) + \mathtt{Coll}_H) - (2\bar{p}+1)^{-\kappa}$. We then define our last game Game_3 as follows:

1. **Run Game_2 once:** $\mathcal{S}$ runs $\mathcal{A}$ by behaving as in Game_2. If $\mathcal{A}$ fails to win the game, then $\mathcal{S}$ aborts. Otherwise, it records $\mathsf{coin}_d, \pi_{K,d}, \mathsf{Chall}_{K,d}, (q_1, q_2, \dots), (h_1, h_2, \dots), i^*$, sets a variable $\mathsf{fork} = 1$ and proceeds to the next step.
2. **(Search for a 3-fork).** This step is repeated twice. $\mathcal{S}$ runs $\mathcal{A}$ with the same random tape as in the beginning of the first step. In addition, it sends $\mathcal{A}$ the same par as before, giving $\mathcal{A}$ oracle access to H_{FS}. $\mathcal{S}$ allows $\mathcal{A}$ to run until termination, answering queries to H_{FS} as follows:
   - Answer queries $q_1, \dots, q_{i^*-1}$ (which are identical to those of the first run) using the same values $h_1, \dots, h_{i^*-1}$ as before.
   - At the i^*-th query q_{i^*} (which is also the same as the first time $\mathcal{A}$ was run), pick a fresh uniform response h'_{i^*}.
   - For the remaining queries made by $\mathcal{A}$ denoted $q'_{i^*+1}, \dots, q'_{Q_H}$, pick fresh uniform random responses $h'_{i^*+1}, \dots, h'_{Q_H}$.

   If this is the first repetition, $\mathcal{S}$ sets $h^{(2)}_{i^*} = h'_{i^*}$. At the second repetition, it sets $h^{(3)}_{i^*} = h'_{i^*}$. If $\mathcal{A}$ terminates without winning the strong exculpability game, then $\mathcal{S}$ begins the next repetition of this step. If $\mathcal{A}$ terminates and wins the game, denote its output as $(PK'_{\mathcal{B}'}, \mathsf{coin}'_1, \mathsf{coin}'_2)$. As before, let $d' \in \{1, 2\}$ denote the index that was not the result of a $\mathcal{Q}_{\mathsf{Spend}}$-query (picking $d' \in \{1, 2\}$ randomly if neither coin was the result of a spend query). Recall that both coins can be the result of $\mathcal{Q}_{\mathsf{Spend}}$-queries with at most negligible probability $\mathsf{negl}(\lambda)_1$, but if this is the case, $\mathcal{S}$ skips to the next repetition of this step. Denote $\mathsf{coin}'_{d'} = (R'_{d'}, \mathbf{y}'_{S,d'}, \mathbf{y}'_{T,d'}, \pi'_{K,d'})$. Write

$$\pi'_{K,d'} = \left(\left\{ \mathsf{Comm}'_{K,d',j} \right\}_{j=1}^{\kappa}, \mathsf{Chall}'_{K,d'}, \left\{ \mathsf{Resp}'_{K,d',j} \right\}_{j=1}^{\kappa} \right).$$

   $\mathcal{S}$ skips to the next repetition of this step at this point if

$$\left(R_d, \mathbf{y}_{S,d}, \mathbf{y}_{T,d}, \{\mathsf{Comm}_{K,d,j}\}_{j=1}^{\kappa} \right) \neq \left(R'_{d'}, \mathbf{y}'_{S,d'}, \mathbf{y}'_{T,d'}, \left\{ \mathsf{Comm}'_{K,d',j} \right\}_{j=1}^{\kappa} \right)$$

   or if $h_{i^*} = h'_{i^*}$. Otherwise, $\mathcal{S}$ sets $\mathsf{fork} \leftarrow \mathsf{fork} + 1$ and $\pi_K^{(\mathsf{fork}+1)} = \pi'_{K,d'}$.
3. **(Derive SIS solution from 3-fork)** If $\mathsf{fork} < 3$ or, $\mathsf{fork} = 3$ but there exists no $j \in [\kappa]$ such that $(h_{i^*}[j], h^{(2)}_{i^*}[j], h^{(3)}_{i^*}[j])$ take three distinct values, then $\mathcal{S}$ terminates outputting $\perp$. Otherwise, $\mathcal{S}$ has access to arguments $\pi_{K,d}, \pi^{(2)}_K, \pi^{(3)}_K$ sharing the same first message which we denote as $\{\mathsf{Comm}_j\}_{j=1}^{\kappa}$. In addition, $\exists j^* \in [\kappa]$ at where $h_{i^*}[j^*], h^{(2)}_{i^*}[j^*], h^{(3)}_{i^*}[j^*]$ take three distinct values in $\{-\bar{p}, \dots, \bar{p}\}$. Now a witness can be extracted from the transcripts $\pi_{K,d}, \pi^{(2)}_K, \pi^{(3)}_K$ by considering the j^*-th parallel repetition and the special-soundness/extractor of the ZKAoK protocol [35]. We denote this witness as $(\bar{J}, \bar{\mathbf{k}}, \bar{\mathbf{e}}_{u^*})$. If $\bar{\mathbf{e}}_{u^*} = \mathbf{e}_{u^*}$, then $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ terminates, outputting $\mathbf{v} := \bar{\mathbf{e}}_{u^*} - \mathbf{e}_{u^*} \in \{-1, 0, 1\}^m$ as a SIS solution.

It then remains to evaluate the probability ϵ_3 that $\mathcal{A}$ succeeds in this last game. We begin by noting that the first and second steps corresponds exactly to the forking algorithm denoted as $F^{\mathcal{A}}$ in Lemma 3. Therefore, a direct application of

this forking lemma implies that the variable fork reaches the value $\mathsf{fork} = 3$ at the beginning of Step 3 with probability at least

$$\mathsf{frk} := \epsilon_2 \cdot \left(\left(\frac{\epsilon_2}{Q_H} \right)^2 - \frac{3}{(2\bar{p}+1)^\kappa} \right).$$

which is non-negligible if ϵ_2 is non-negligible as $1/(2\bar{p}+1)^\kappa$ is negligible and Q_H is polynomial. Next, note that $\mathcal{S}$ extracts a witness $(\bar{J}, \bar{\mathbf{k}}, \bar{\mathbf{e}}_{u^*})$ if and only if it does not terminate at, or before the beginning of Step 3. In order to analyse the probability that this occurs, we define three events:

- GF ("Good fork"): This is the event that $\mathsf{fork} = 3$ and there exists an index $j^* \in [\kappa]$ such that $(h_{i^*}[j^*], h_{i^*}^{(2)}[j^*], h_{i^*}^{(3)}[j^*])$ is a triple of 3.
- F ("Any fork"): This is the event that $\mathsf{fork} = 3$ at the beginning of Step 4.
- GH ("Good hashes"): This is the event that there is an index $j^* \in [\kappa]$ such that $(h_{i^*}[j^*], h_{i^*}^{(2)}[j^*], h_{i^*}^{(3)}[j^*])$ take 3 distinct values.

It is easy to see that $\Pr[\overline{\mathsf{GH}}] = ((6\bar{p}+1)/(2\bar{p}+1)^2)^\kappa$ is negligible and that $\Pr[\mathsf{F}] = \mathsf{frk}$. We also have

$$\Pr[\mathsf{F}] \leq \Pr[\mathsf{F}|\mathsf{GH}] \cdot 1 + 1 \cdot \Pr[\overline{\mathsf{GH}}] = \Pr[\mathsf{F}|\mathsf{GH}] + \mathsf{negl}(\lambda).$$

This implies that $\mathcal{S}$ does not abort at the beginning of Step 3 or before with non-negligible probability

$$\Pr[\mathsf{GF}] = \Pr[\mathsf{F} \cap \mathsf{GH}] = \Pr[\mathsf{F}|\mathsf{GH}] \cdot \Pr[\mathsf{GH}] \geq (\mathsf{frk} - \mathsf{negl}(\lambda)) \cdot (1 - \mathsf{negl}(\lambda)).$$

The last step is to evaluate the probability that $\bar{\mathbf{e}}_{u^*} = \mathbf{e}_{u^*}$, leading $\mathcal{S}$ to abort. Here we rely on our previous observation, namely that the adversary's view has been independent of $\mathbf{e}_{u^*}$ since $\mathsf{Game}_{1,Q_w+Q_s}$ and that there is, with overwhelming probability, at least another vector $\bar{\mathbf{e}}_{u^*} \neq \mathbf{e}_{u^*}$ that is a valid secret key for $PK_{\mathcal{U}^*}$. We therefore know that the probability of the event $\bar{\mathbf{e}}_{u^*} \neq \mathbf{e}_{u^*}$ is at least $\frac{1}{2}$. In summary, we get the following bound on the probability ϵ_3 that $\mathcal{A}$ succeeds in Game_3:

$$\epsilon_3 \geq \frac{1}{2} \cdot (\mathsf{frk} - \mathsf{negl}(\lambda)) \cdot (1 - \mathsf{negl}(\lambda))$$

where frk is defined above. Any adversary $\mathcal{A}$ succeeding with non-negligible probability ϵ against the exculpability of our scheme can thus be used to solve the SIS problem, distinguish the BLMR PRF from pseudo-random, or break the zero-knowledge property of Π_1 or Π_2, which completes the proof. □

7 A More Efficient GGM-based Construction

In Section 4, we use the BLMR PRF because it allows for a simpler description of the argument of knowledge, as it only requires one rounding per evaluation. Unfortunately, this comes at the price of a super-polynomial modulus q. We

can do better by using a PRF obtained by applying the seminal construction of Goldreich, Goldwasser and Micali [23] to the LWR-based PRG of Banerjee *et al.* [4] for which the LWE-to-LWR reduction of [2] allows the use of a polynomial modulus. This leads to an e-cash construction with $q = \mathsf{poly}(\lambda)$ which still relies on the hardness of standard worst-case lattice problems. Explicitly, the PRF we have in mind relies on the hardness of the $\mathsf{LWR}_{m,m,q,p}$ problem (which is at least as hard as $\mathsf{LWE}_{m',m,q,\alpha'}$ for $m' \geq \frac{\log q}{\log(2\gamma')}m, q \geq \gamma' m^2 \alpha' p$ for any $\gamma' \geq 1$ [2]). This PRF uses public parameters $m, p, q, \mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{m\times m}$ where $\mathbf{A}_0, \mathbf{A}_1 \hookleftarrow U(\mathbb{Z}_q^{m\times m})$. The evaluation on seed $\mathbf{k} \in \mathbb{Z}_q^m$ and input $x \in \{0,1\}^L$ is

$$F_{\mathbf{k}}(x) := \left\lfloor \mathbf{A}_{x_L} \cdot \left\lfloor \ldots\ldots \left\lfloor \mathbf{A}_{x_2} \cdot \lfloor \mathbf{A}_{x_1} \cdot \mathbf{k} \rfloor_p \right\rfloor_p \ldots\ldots \right\rfloor_p \right\rfloor_p . \tag{4}$$

When replacing the BLMR PRF with the above in our e-cash construction, it is more convenient to keep the parameters m and n as described in Section 4. This allows us to reuse our security proofs without any issues. However, in contrast with the BLMR instantiation, we choose polynomially large p and q such that $q^2 > m^{5/2}p$ in the $\mathsf{ParGen}()$ phase. In addition, the binary public matrices $\mathbf{P}_0, \mathbf{P}_1$ must be replaced by uniformly sampled $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{m\times m}$. In the full version [16], we show that this alternative PRF is compatible with the ZK relation $\mathcal{R}^*$ considered in [35], as we did for the BLMR PRF in Section 5.1. Combining this with the reasoning in Section 5.2 allows us to show that the GGM-based PRF is compatible with the ZKAoKs used in Spend.

7.1 Parameters

We provide in this section some details on the parameters and the complexity of an instantiation of our e-cash system using the GGM-based PRF. Firstly, Theorem 1 states that the security of our construction relies on:

- $\mathsf{LWR}_{m,m,q,p}$ (which is at least as hard as $\mathsf{LWE}_{m',m,q,\alpha'}$ for $m' \geq \frac{\log q}{\log(2\gamma')}m, q \geq \gamma' m^2 \alpha' p$ for any $\gamma' \geq 1$ [2])
- $\mathsf{LWE}_{n_{\mathsf{LTF}},m_{\mathsf{LTF}},q_{\mathsf{LTF}},\alpha}$ with $\alpha = \Theta\left(\frac{\sqrt{n_{\mathsf{LTF}}}}{q_{\mathsf{LTF}}}\right), q_{\mathsf{LTF}} = \Theta(n_{\mathsf{LTF}}^{1+1/\gamma})$ for constant $\gamma < 1$
- $\mathsf{SIS}_{n,m,p,2\sqrt{m}}$
- $\mathsf{SIS}_{n,m_s,q_s,\beta'}$ for $\beta' = \mathcal{O}(\sigma^2 m_s^{1/2}(m_s + \bar{m}))$

and also that we use secure ZKAoKs. Since all moduli will be polynomial, we may safely assume that there is a parameter setting such that the argument system of Yang *et al.* is a ZKAoK. Additionally, our proof of the clearing property requires use of a signature scheme. Note that we can use the signature scheme of [22] so that the arising assumption is made redundant by the final item listed above. Recall that for our zero-knowledge proofs, we require that q_s, q_{LTF} and p all divide the prime power q. In order to achieve this, we now set $q = q_0^e$ where q_0 is prime and $e > 1$ is a constant integer. Since all moduli are polynomial, we may take $n_{\mathsf{LTF}} = \Theta(m) = \Theta(n \log q) = \tilde{\mathcal{O}}(n)$. Additionally, $m, \bar{m}, m_s, m_{\mathsf{LTF}}, \bar{n}_{\mathsf{LTF}}$ and n'

are all $\tilde{\mathcal{O}}(n)$. Note that we will take $\gamma' = 1$ in the LWE-to-LWR reduction result stated above and $\gamma = 1/2$. To comply with hardness results relating standard worst-case lattice problems to SIS [22,29] and LWE [31,10], we require:

$$q^2/p = \tilde{\Omega}(n^{5/2}) \qquad q_{\mathsf{LTF}} = \tilde{\Theta}(n^3) \qquad p = \tilde{\Omega}(n) \qquad q_s = \tilde{\Omega}(\sigma^2 n^2) = \tilde{\Omega}(n^3).$$

Therefore, to base security on worst-case lattice problems, we may take n,m,n_{LTF}, $\bar{n}_{\mathsf{LTF}}$,m_{LTF},m_s all $\tilde{\mathcal{O}}(\lambda)$, $p = q_0 = \tilde{\mathcal{O}}(\lambda)$ and $q = q_s = q_{\mathsf{LTF}} = q_0^3 = \tilde{\mathcal{O}}(\lambda^3)$. Additional details on the communication costs are provided in the full version of this work.

Acknowledgements

This work is supported by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701) and also partly funded by BPI-France in the context of the national project RISQ (P141580). Khoa Nguyen is supported in part by the Gopalakrishnan - NTU PPF 2018, by A*STAR, Singapore under research grant SERC A19E3b0099, and by Vietnam National University HoChiMinh City (VNU-HCM) under grant number NCM2019-18-01.

References

1. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *Eurocrypt*, 2010.
2. J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs. Learning with rounding revisited – new reduction, properties and applications. In *Crypto*, 2013.
3. A. Banerjee and C. Peikert. New and improved key-homomorphic pseudo-random functions. In *Crypto*, 2014.
4. A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In *Eurocrypt*, 2012.
5. C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert. More efficient commitments from structured lattice assumptions. In *SCN*, 2018.
6. M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In *Pairing*, 2009.
7. D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. Key-homomorphic prfs and their applications. In *Crypto*, 2013.
8. J. Bootle, V. Lyubashevsky, and G. Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *Crypto*, 2019.
9. F. Bourse, D. Pointcheval, and O. Sanders. Divisible e-cash from constrained pseudo-random functions. In *Asiacrypt*, 2019.
10. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. On the classical hardness of learning with errors. In *STOC*, 2013.
11. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *Eurocrypt*, 2005.
12. S. Canard, D. Pointcheval, O. Sanders, and J. Traoré. Divisible e-cash made practical. In *PKC*, 2015.

13. D. Chaum. Blind signatures for untraceable payments. In *Crypto*, 1982.
14. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *Eurocrypt*, 2000.
15. Observatoire de l'épargne réglementée. Rapport annuel. `https://www.banque-france.fr/sites/default/files/medias/documents/observatoire-de-l-epargne-reglementee-rapport_2013.pdf`, 2013.
16. A. Deo, B. Libert, K. Nguyen, and O. Sanders. Lattice-based e-cash, revisited (full version). *IACR Cryptol. ePrint Arch.*, 614, 2020.
17. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *SIAM*, 2008.
18. A. El Kaafarani and S. Katsumata. Attribute-based signatures for unbounded circuits in the rom and efficient instantiations from lattices. In *PKC*, 2018.
19. M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu. Matrict: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In *ACM CCS*, 2019.
20. P. Farshim, C. Orlandi, and R. Rosie. Security of symmetric primitives under incorrect usage of keys. In *ToSC*, 2017.
21. A. Fiat and A. Shamir. How to prove yourself – practical solutions to identification and signature problems. In *Crypto*, 1986.
22. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.
23. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. In *FOCS*, 1984.
24. S. Kim and D. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In *Crypto*, 2017.
25. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *Asiacrypt*, 2016.
26. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based prfs and applications to e-cash. In *Asiacrypt*, 2017.
27. V. Lyubashevsky. Lattice-Based Identification Schemes Secure Under Active Attacks. In *PKC*, 2008.
28. Mastercard. Transaction processing rules. `https://www.mastercard.us/content/dam/mccom/global/documents/transaction-processing-rules.pdf`, 2019.
29. D. Micciancio and C. Peikert. Hardness of SIS and LWE with small parameters. In *Crypto*, 2013.
30. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *STOC*, 2008.
31. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.
32. D. Ron and A. Shamir. Quantitative analysis of the full bitcoin transaction graph. In *Financial Cryptography*, 2013.
33. Visa. Transaction acceptance device guide. `https://www.visa.com.pe/dam/VCOM/regional/na/us/partner-with-us/documents/transaction-acceptance-device-guide-tadg.pdf`, 2016.
34. H. Wee. Dual projective hashing and its applications — lossy trapdoor functions and more. In *Eurocrypt*, 2012.
35. R. Yang, M.H. Au, Z. Zhang, Q. Xu, Z. Yu, and W. Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Constructions and applications. In *Crypto*, 2019.